Knowledge-based 3D point clouds processing

Quoc Hung Truong

SPIM

Thèse de Doctorat

école doctorale **sciences pour l'ingénieur et microtechniques**

UNIVERSITÉ DE BOURGOGNE

Knowledge-based 3D point clouds processing

Traitement 3D de nuages de points basé sur la connaissance

Quoc Hung TRUONG

Thèse de Doctorat

N° | X | X | X |

THÈSE présentée par

Quoc Hung TRUONG

pour obtenir le

Grade de Docteur de
l'Université de Bourgogne

Spécialité : **Informatique**

Knowledge-based 3D point clouds processing

Traitement 3D de nuages de points basé sur la connaissance

Soutenue le 15 November 2013 devant le Jury :

Prof. Laurent BIGUE	President	Université de Mulhouse
Prof. Yvon VOISIN	Director	Université de Bourgogne
Prof. Frank BOOCHS	Co-director	Fachhochschule Mainz
Prof. Adlane HABED	Co-supervisor	Université de Strasbourg
Prof. Sylvie TREUILLET	Reviewer	Université d'Orléans
Prof. Christophe DOIGNON	Reviewer	Université de Strasbourg

ACKNOWLEDGEMENT

I would here like to express my thanks to the people who have supported me during the PhD study.

My first debt of gratitude must go to my supervisor, Prof. Yvon Voisin, who gives me the chance to pursue a PhD as well as supports to carry out my thesis at the Laboratory of Electronics, Computer science and Image (LE2I). I would like to express my sincerest thanks and appreciation to my second supervisor, Prof. Adlane Habed. He provided the vision, encouragement and advise necessary for me to proceed through the doctoral program and complete my dissertation.

I am deeply indebted to Prof. Frank Boochs, my thesis would not have come to a successful completion without the support, encouragement and invaluable suggestions I received from him. He has enabled my research work during my PhD and it has been a great privilege to spend several years in i3mainz.

I would specially like to thank Dr. Ashish Karmacharya and members of i3mainz for their support, guidance and helpful suggestions. Their guidance has served me well and I owe them my heartfelt appreciation.

The best and worst moments of my doctoral journey have been shared with my family. I wish to thank my family whose love and encouragement allowed me to finish this journey.

Contents

1

Introduction

1.1/ Context and Motivation

Three-dimensional (3D) point cloud processing has lately known a growing interest following a surge in scanning technologies and capabilities. As collecting and digitizing 3D data from the real-world have become readily accessible, several applications – whether in industry, security, robotics, or even in the medical field – have already adopted 3D scanning as an indispensable tool. While laser scanners have already been established as a workhorse for topographic and building surveys, the introduction on the market of affordable and simple scanning devices (such as Microsoft's Kinect), 3D scanning is expected to reach an all new level of proliferation. With every new scanner model on the market, phase-shift scanners in particular, such instruments are becoming faster, more accurate and can scan objects at longer distances. However, increasing scanning speed leads to a new behaviour of the user in the field. While region to be scanned and equipment resolution have to be chosen carefully when using slow pulse scanners, phase-shift scanners usually carry out high resolution scans exhibiting great redundancy and density of points. In most cases, processing techniques are still mainly relying on user intervention. Typical operations consist in denoising, deleting unnecessary areas, navigating in an often huge and complicated 3D structure, selection of sets of points, extraction and modelling of geometries and objects, etc. Such tasks are tedious and generally require a high level of expertise as well as a lengthy training of personnel.

The development of processing algorithms in point clouds, such as registration, noise reduction, feature extraction, model fitting, object detection, etc, has been a key concern in the research areas of Computer Vision and Photogrammetry. Object detection and reconstruction from digitized data, typically images and point clouds, are important tasks that find applications in many fields. Because such processing tasks are extremely laborious and difficult when carried out manually, it is of the utmost importance that they benefit from the support – or even be entirely performed through – numerical algorithms. Most existing 3D processing techniques and object detection methods are data-driven. For instance, several methods proceed by fitting models with scans. Typically, these methods proceed by segmenting the point cloud under consideration and measuring the similarity between the model and the extracted features. Some methods rely both on extracting discriminating features from the data set as well as on numerical models characterizing either geometric (e.g. flatness and roughness) or physical (e.g. color and texture) properties of the sought objects. The numerical model and the extracted features are combined to form a decision. Other methods employ object segmentation techniques

and classification, possibly through learning. While using learning has the drawback of requiring application-dependent training sets (often difficult to obtain), those methods that do not rely on learning lack the flexibility expected from such systems. Indeed, the latter methods are applied in a static manner regardless of the context or any scene-dependent knowledge that might be available. These approaches often have one thing in common: they are static and do not allow a dynamic adjustment to the object or to the initial processing results. The employed algorithm is applied to the entire point cloud and the result can be good or bad. The result is dependent upon several factors such as point cloud quality, object distribution in a scene, object features and so on. However, there is no feedback to the algorithmic part in order to either activate a different algorithm or use the same algorithm with revised parameters. This interaction is still up to the users who have to decide which algorithms are to be applied for the kind of objects and point clouds at hand. This again leads to a time-consuming mostly manual process.

With the increasing complexity of the data and the objects represented therein, a correct validation of the numerically modeled features becomes increasingly difficult and renders decisions based on individual algorithmic features unreliable. This problem can be solved by taking into account additional guiding information within the algorithmic process chain as to support the validation process. Such information might be derived from the context of the object itself and its behavior with respect to the data and/or other objects or from a systematic characterization of the parameterization and effectiveness of the algorithms to be used. Indeed, most conventional methods are generally affected by the nature of data set and the behavior of the algorithms. It is up to the user to decide, often subjectively but generally based on one's experience, which algorithms are best suited for any particular kind of objects and/or data sets. It goes without saying that the success of these approaches is significantly compromised by the increasing complexity of the objects and the decreasing quality of the data. Furthermore, relying on only a restricted set of features and individual algorithms to process the data might lead to unreliable results.

However, as far as the 3D processing of the resulting digitized scenes is concerned, this seems to remain a matter that is limited to "knowledgeable" expert individuals, hence not accessible to many. One way to overcome the drawbacks of the data-driven approaches is to resort to the use of additional knowledge. For instance, knowledge characterizing the objects to be detected with respect to the data at hand, or their relationships to other objects, may generally be derived beforehand. Such knowledge not only allows for a systematic characterization and parameterization of the objects but also supports the quantification of the effectiveness of the algorithms to be used. Methods combining data-driven processing algorithms and semantic knowledge have been proposed in the literature (these are discussed in the early chapters of this thesis). Such combination has not only led to faster 3D processing of point clouds, in particular when dealing with large sets of data, but also to more "intelligent" strategies for detecting objects and annotating scenes. For example, semantic knowledge has been used to support the building of semantic maps for autonomous robots from laser scans. Also, a combination of semantic 3D object maps and triangulated surface maps has been used to allow a personal robotic assistant to classify regions and estimate 3D geometrical features.

A limitation that is common to all methods relying on the support of knowledge to detect objects and annotate scenes is that their algorithms act on the results of the data-driven segmentation/processing and do not exploit semantic knowledge to guide and direct the processing itself. Indeed, the results of the 3D processing algorithms are very much dependent upon the quality of the scanned data and on the topology of the scene be-

ing scanned. When a 3D processing task fails to provide adequate results, reasoning - based on the knowledge being used and the result of the processing - necessarily fails as well. The 3D processing tasks mostly fail however because they are unable to adapt to particular circumstances. Therefore, it becomes necessary to devise a new approach that supports 3D processing with knowledge, allows guiding, controlling and adapting the processing of a point cloud in a continuous situation-specific manner. Additionally, this approach should optimize the 3D processing by dynamically and automatically selecting suitable algorithms based on both knowledge and feedback from the processing results to decide for the next processing step. A platform integrating knowledge and 3D processing to detect and annotate point clouds is therefore required.

1.2/ SCOPE OF THE THESIS

Problem statement

Digitized realistic 3D data have proven useful in a variety of industrial applications, ranging from security and robotics to healthcare and surgical support. In this thesis, we focus on investigating the problems of detecting and identifying objects laser scanned data, typically point clouds, in site surveying applications. The approach is to devise a robust numerical processing approach, employing point cloud preprocessing, feature extraction, geometry fitting, etc., and providing reliable results under different conditions of data and scene. In particular, the proposed approach relies on knowledge to guide processing algorithms in the task of detecting and identifying objects in 3D point clouds. The algorithms are combined in a flexible manner to act on data while relying on knowledge about objects' characteristics and relationships between them. In addition, our goal is also to include in this approach the automatic selection of algorithms as to detect objects by reasoning upon situations and analyzing relevant knowledge.

The work presented in this thesis uses semantic knowledge and employs a cognitive approach to guide the processing. This is motivated by the need to replace the current pure-numeric strategies by fault-tolerant and adaptive methods for object extraction and identification, which are modeled in the knowledge domain. Data sources like images, stereo pairs or point cloud colour information can be included in the detection process. In contrast to existing approaches, we aim at utilizing previous knowledge on the objects to measure. This knowledge can be contained in databases, construction plans, as-built plans, Geographic Information Systems (GIS) or just obtained from domain experts. Such knowledge is the basis for a selective, object-oriented detection, identification and, if necessary, modelling of the objects and elements of interest in the point clouds.

Solution

The work proposed in this thesis bridges between semantic modeling and numerical processing strategies. This avoids actual limits in the use of knowledge within numerical strategies. As a basis for our approach, available knowledge is structured and explicitly formulated by linking objects geometry to semantic information, creating rules and guiding the algorithms used to process the real data. The general process architecture of our system consists of two distinct parts that are combined seamlessly to carry out the knowledge-based operations during the processing steps.

The first part encapsulates knowledge through the semantic definitions of the domains that are involved. Four knowledge domains are involved in this process and should be identified in all scenarios by human observation (for example through the scanning of documents, site plans, CAD drawings, and GIS). For instance, we relate data set knowledge and expert knowledge about processing algorithms to the geometrical or topological behavior of the object. The created knowledge is structured into an ontology containing a variety of elements such as prior information about the objects extracted from data sources (digital maps, geographical information systems, etc) or information about the objects' characteristics, a hierarchy of the sub-elements, the geometrical topology, the characteristics of the processing algorithms etc. During processing, such modeled knowledge provides relevant information allowing for the guidance of the analysis and the identification processes. This allows choosing from different algorithmic strategies, possibly combining them and reacting to "unexpected" situations by making use of the overall knowledge framework. To achieve this, all relevant information about the objects, the algorithms and their interrelationships ought to be modeled inside the ontology, including characteristics like positions, geometric descriptions, texture images, behavior and parameters of suitable algorithms, etc.

If knowledge is considered as a critical element in guiding the various data processing stages, the processing algorithms, which account for the second part of our system, play an important role in detecting geometries in point clouds. The algorithmic part includes a number of algorithms such as noise reduction in point clouds, the removal of outliers, data partitioning, segmentation, geometry fitting, etc. These algorithms are independent components in the sense that each can function independently from the others. However, they could also be combined to create a sequence that allows the detection of geometries present within the object. Note that, with a large variety of object types of diverse complexity, a collection of many algorithms is needed in practice. In order to manage these algorithms, we propose to classify them into individual groups according to the task they have been designed for. This structure allows making the algorithms readily available for easy access under the guidance of knowledge as to direct, adapt and select the most suitable algorithms based on the objects characteristics as well as to adjust their parameters to the current situation. The characteristics are considered as values that can change the parameters of the algorithms thus adapting to current conditions. After an object is detected, its status is fed back into the knowledge part only to be taken into account in subsequent processing stages.

The selection of an appropriate sequence of algorithms is carried out through the so-called Algorithm Selection Module (ASM). This module takes expert knowledge on processing into account and combines it with domain knowledge in order to support appropriate algorithm selection for every particular case. Each algorithm behaves differently in combination with other algorithms. All algorithm characteristics and relations to other algorithms should be taken into account while creating any chain that combines an algorithm sequence. This knowledge is based on empirical studies and simulations carried out by domain professionals and mapped on the knowledge schema. Using the modeled algorithm characteristics, a graph representing all possible travel directions is created and helps determining the appropriate flow of algorithm sequences.

1.3/ CONTRIBUTIONS

The work presented in this thesis aims at efficiently exploiting additional knowledge in the processing of point clouds. Our main contributions can be stated as follows:

- We present a comprehensive review of state-of-the-art methods in both 3D data processing and knowledge-based systems.

- We propose a framework to model and use knowledge from various domains and to make it contribute to all steps of an object detection process. This starts with inferring the steps that control algorithms based on object and scene-related knowledge (in order to select appropriate algorithmic strategies) and ends with a knowledge-based object classification while simultaneously extending and updating the knowledge base Boochs et al. [2011], Truong et al. [2013a].

- We also propose a structured organization of a number of numerical processing algorithms that serve as a basis for tasks such as data preprocessing, segmentation, geometry fitting, etc. This work has been published in Truong et al. [2010], Marbs et al. [2010].

- We bridge semantic knowledge (taken from multiple sources such as digital maps and geographical information systems) and numerical processing strategies in order to benefit from knowledge in any or all parts of an automatic processing chain Truong et al. [2012]. This approach not only relies on information about potentially present objects in the scene (their characteristics, a hierarchal description of their sub-components, spatial relationships) but also on the characteristics of the processing algorithms at hand. During processing, the modeled knowledge guides the algorithms and supports both the analysis of the results and the object classification. Knowledge is also used to support the choice among different algorithms, the combination of these, and the adopted strategies Truong et al. [2013b].

- We have developed a demonstration prototype “Wissensbasierte Detektion von Objekten in Punktwolken für Anwendungen im Ingenieurbereich” (WiDOP) of our knowledge-driven approach. The solution rests on some fundamental knowledge domains: scene knowledge, data knowledge, spatial knowledge and algorithm knowledge. These domains allow describing the scene and the semantic behavior of the processing algorithms. The semantic knowledge used to relate these domains provides the much needed flexibility in the algorithmic processing. Our demonstrator uses datasets provided by the Deutsche Bahn AG (German Railway system) and Fraport AG (Frankfurt International Airport) to show the effectiveness of the approach.

1.4/ THESIS OVERVIEW

The thesis is structured as follows. An overview of the relevant literature on 3D point cloud processing and that of knowledge-based systems is given in Chapter 2. Chapter 3 covers the necessary background used in this thesis. We introduce aspects of semantic knowledge engineering as well as its usage in the scope of our work. The fundamental algorithms in image processing and 3D point cloud processing are highlighted in this chapter as well. Chapter 4 is dedicated to our knowledge-based strategy which addresses the problem of 3D object detection and classification in point clouds. The approach iden-

tifies several knowledge domains as to devise an algorithm selection module that enables us to automatically identify objects in the scanned data. This is followed by case-studies involving real-world examples in Chapter 5. Both cases, one for an indoor situation and the other for an outdoor scene, require robust methods to detect and classify objects in 3D point clouds. Our approach was tested in these two scenarios and achieved reliable results. Our conclusion and future work are given in Chapter 6.

2

LITERATURE REVIEW

In this chapter, we present a comprehensive review of data processing methods. The discussed methods are classified into five main groups according to the processing paradigm they are based upon. For instance, we distinguish purely model-driven and data-driven approaches from those paradigms incorporating intelligence and semantic aspects in the processing. We also discuss processing methods that are based on various forms of knowledge and which represent the basis of the approach we propose and defend in the present thesis.

2.1/ MODEL-DRIVEN APPROACHES

Model-driven approaches resort to the design of a model consisting in a mathematical or graphical description the sough object. The model serves as a reference against which the data being processed is compared. Several methods based on this approach have been developed to solve a number of problems particularly in the field of Computer Vision and of which the most relevant ones are discussed here.

[Kragic and Christensen 2002] proposed the use of a model-based tracking system to estimate and continuously update the pose of an object to be manipulated. A wire–frame model is employed to identify and track features across images. One of the important parts of their system is the ability to automatically initiate the tracking process and operate in a domestic environment with changing lighting and background conditions. In [Truong et al. 2008] a model-based 3D object recognition method, employing intersecting lines and a pre-defined object model, has been proposed. 3D line-segments are extracted using both 2D images and point clouds yielding the identification of pairs of interest lines with given angle. By estimating the coverage ratio, the algorithm finds the most accurate matching between detected line pairs and a model database. Note that several model-based methods proceed in a way that is similar to the above two approaches. For instance, objects are only described in terms of their shape using a wire-frame model without incorporating any further details and features such as corners, interest points, color or texture.

Other methods making use of such features in addition to a model have also been proposed in the literature. For instance, the authors of [Taylor and Kleeman 2003] have developed a fusion scheme for 3D model-based tracking using a Kalman filter framework. Color, edge and texture cues, predicted from a textured CAD model of the tracked object,

have been used to recover the 3D pose. Such approach has also the ability of taking additional cues and cameras into account within the tracking algorithm provided a suitable measurement function exists. In term of changes in visual conditions, this approach outperforms methods employing single-cue algorithms. In [Ekvall et al. 2005], the authors propose another approach for object recognition and pose estimation that is based on color co-occurrence histograms and geometric modeling. This method employs a classical learning framework and color co-occurrence histograms that facilitate a "winner-takes-all" strategy across different views and scales. The hypotheses generated in the recognition stage provide the basis for estimating the orientation of the object around the vertical axis. The system can automatically initiate an object tracking process. It uses either recognition or pose estimation, both relying on the same object representation. This approach yields a gain in robustness, invariance with respect to scaling and translation as well as computational efficiency. The major contribution of their work is in the integration of different techniques to obtain real-time, on-line 6DOF pose estimation, one of the few systems that are able to perform automatic initialization of the pose tracking algorithm.

Model-based object recognition approaches utilize the knowledge of an object's appearance that is provided by an explicit model of its shape. Such techniques not only recognize objects through representing shape but also fuse other additional properties. The advantage of using Model-driven approaches particularly manifests in the presence of low quality data exhibiting a lack of object representation. However, such approaches suffer from some obvious limitations in particular when the shape of the object is particularly complicated. In such case the object's representation may requires a faithful description that may not be easy to obtain. Another shortcoming of the model-based approaches has to do with the effect of noise on the mapping between the data and the model which could very well lead to failure.

2.2/ PURELY DATA-DRIVEN APPROACHES

In contrast to model-driven approaches, data-driven ones act on the data without resorting to the use of any predefined model of the sought object regardless of its form. Such approaches proceed by extracting primitive features, should they be simple or complex geometric features, from visual information such as point cloud data. The extracted features are combined in such a way a model, not defined beforehand, is generated. Early 3D processing techniques were purely data-driven exhibiting obvious limitations with the increasing complexity of the data and scenes. Progress has been achieved by considering the use of models approximating geometrical characteristics of objects. However, despite the robustness and efficiency of many such processing algorithms, they alone cannot resolve existing ambiguities when qualifying objects in a digitized scene. Such ambiguities can be efficiently dealt with when integrating semantic knowledge with data processing.

As far as feature-based object recognition is concerned, some approaches have been used in both 2D images and 3D data. Many of these were dedicated to the reconstruction of buildings. For instance, [Vosselman and Dijkman 2001] made use of higher level 3D features, usually simple roof shapes (flat roofs, gable roofs and hip roofs) that are generally present in building structures. The authors relied on the use of the 3D Hough transform to detect planar roof faces in point clouds, and hence reconstructed the scene in a higher level of abstraction. Their segmentation strategy was based on detecting

intersecting lines and “height jump edges” between planar faces. In general, such methods require some level of user intervention. The user manually initializes the process by providing some measurements based on which an algorithm attempts to extract other elements. These methods are usually supported by (orthogonal or perspective) projections [Zitova and Flusser 2003] in a lower dimensional space as to make some constraints more evident.

When modeling buildings by constructive solid geometry, buildings can be regarded as compositions of a few components with simple roof shapes, like flat roofs, gable roofs and hip roofs. This has motivated the work [Bredif et al. 2007] and the one in [Lafarge et al. 2008] for developing point cloud segmentation methods specifically designed for buildings and relying on similarity measurements between the model and the extracted features. For instance, planes are very useful features as they are typically present in any man-made environments. For example, [Ameri and Fritsch 2000] introduces a method for automatic 3D building reconstruction using plane-roof structures. First, the system construct a boundary representation for a coarse building hypothesis based on a bottom-up approach starting from simple geometric primitives (projections of co-planar feature roofpoints or lines), that are present in images, to complex geometric primitives (for instance a roof structure) in the scene. Subsequently, the top-down approach is applied to back project the reconstructed model to the corresponding images for verifying the hypothesis model.

A stricking example of a data-driven approach is the one in [Pollefeys et al. 2000] where the authors show the feasibility of a 3D reconstruction from a hand-held camera without prior knowledge neither about the scene nor about the camera. The authors proposed a system which automatically extracts a textured 3D surface model from a sequence of images of a scene. Building the 3D model is processed without camera settings and scene knowledge. The system uses recently developed Computer Vision algorithms and 3D modeling tasks decomposed into several consecutive steps. The scene and camera settings are then gradually retrieved. The obtained accuracy is not yet at the level required for most metrology applications, but the visual quality is very convincing. This approach has been applied to a number of applications in archaeology.

As far as point cloud data are concerned, typically obtained from laser scanners, [Alharthy and Bethel 2004] shows that dense airborne laser scanning data may suffice for a detailed 3D reconstruction of urban features such as buildings. Local statistical inference is used and least-squares analysis of moving surfaces has been critical in determining building roof details. The consistency of the data with those surfaces determines how they can be modeled. Complete wireframe of buildings is constructed after obtaining the roof facet orientation and approximate location, then, from the intersection of these facets, the roof boundary is extracted. The method presented in [Pu and Vosselman 2006] used segmentation and feature extraction algorithms to recognize building components (such as doors, walls, windows) from point clouds. Based on constraints on the sought components, they were able to determine the categories each extracted feature belonged to. However, the results of these methods were not satisfying when the data did not clearly describe the object due to either the presence of noise or because of occlusions. In fact, the visual information (images, point clouds...) which is acquired from different scanners has different property and quality, such as density of point clouds. The nature of scanned area also determines complexity of the scene and object. Feature-based object recognition methods depend on the representation of the objects in the data. Their outcome is very much dependent upon the accuracy from the processing algorithms, such

as segmentation, extraction and model fitting. This is a recurrent issue because each processing algorithm comes with its own limitation and is only able to work effectively under certain conditions and hypotheses.

2.3/ INTELLIGENT DATA-DRIVEN APPROACHES

Additional aspects need to be taken into account in order to cope with the limitations of the conventional purely data-driven approaches in term of uncertainty and complexity of data. Methods have been proposed to improve the existing data-driven approaches by using concepts from machine learning as to enforce the robustness of such methods in recognizing and processing complex objects and scenes. Machine learning is traditionally considered as part of the field of Artificial Intelligence and aims at building programs whose behavior changes (and improves) through experience or use a learning process. Machine learning is integrated into data-driven approaches to solve the two most important problems: classification and regression (numerical prediction). Classification methods may employ decision trees, Bayesian methods, instance-based learning (k-nearest neighbor algorithm) and self-organizing feature maps and many other techniques.

A typical work in this category is the one presented by [Anguelov et al. 2005] in which object segmentation and classification are obtained through a learning procedure employing Markov Random Fields (MRFs) and quadratic programming. The MRF models incorporate diverse features and enforce the preference that adjacent scan points have the same classification label. Maximum margin framework is proposed to discriminatively train the model from a set of labeled scans. Finally, the system ends-up automatically learning the relative importance of the features for the segmentation task.

Another method worth mentioning in the same category is the one proposed by [Triebel et al. 2007b] which classifies more complex objects based on a diverse set of features. By using the distances of features to their nearest neighbors, the transformed feature space becomes more easily linearly separable. The associative Markov networks (AMNs) is incorporated within the framework to improve the performance of the training step. However, the drawback of this approach is that, by storing instances, the resulting classifier becomes a lazy classification method. The inference step in this approach requires computing the distance between the instance to be classified and the known training instances. To overcome the computational issues that arise in such calculations, data is structured and the system uses kD-Trees to improve the performance.

[Golovinskiy et al. 2009] investigate a system for recognizing objects in 3D point clouds of urban environments. The system consists of four steps: locating, segmenting, characterizing, and classifying clusters of data. After locating the potential object positions by clustering nearby points, the system segments points near those positions. Each point cluster has a feature vector and the feature vectors are labeled using a classifier trained on a set of manually labeled objects.

Such methods, however, generally require a large number of training data sets in order to obtain good results. While using learning has the drawback of requiring application-dependent training sets (often difficult to obtain), those methods that do not rely on learning lack the flexibility expected from such systems. Indeed, the latter methods are applied in a static manner regardless of the context or any scene-dependent knowledge that might be available.

2.4/ DATA-DRIVEN INCORPORATING SEMANTICS

Some approaches rely on a combination between data-driven processing algorithms and semantic knowledge. This has not only lead to faster 3D processing of point clouds, in particular when dealing with large sets of data, but also to more “intelligent” strategies of detecting objects and annotating scenes. Unlike methods based on Machine Learning, incorporating semantic knowledge requires no learning step. Taking semantic knowledge into account has brought significant improvements to the processing of 3D data as demonstrated by the results reported [Duan et al. 2010] for the automatic data extraction process from 3D point clouds.

The early method proposed by [Cantzler et al. 2002] relies on a semantic network defining the relationships among objects in a scene (such as walls being perpendicular to the floor) and rules which the extracted features must obey. The interesting issues come however with complex indoor scenes, including many types of objects. [Hedau et al. 2009] recovered spatial layout of indoor scenes by modeling the global room space with a parametric 3D box before iteratively localizing clutter and refitting the box. In a similar approach, [Lee et al. 2010] parametrically represented the 3D volume of objects and rooms that allowed them to apply constraints for volumetric reasoning, such as spatial exclusion and containment.

Later, [Hedau et al. 2010] located objects of a specific geometry in an indoor scene. Using object geometry, scene geometry, their mutual arrangement and a single image, the detector computes object location in 3D along with its orientation. These works aided in scene understanding considering, however, a single object in the scene. Localizing multiple objects in a scene remains a difficult problem for which no reliable solution exists. One way to address this problem is to resort to the use of semantic knowledge. Semantic knowledge is defined in terms of geometric constraints [Nuechter et al. 2006]. This has turned out to be very useful in building indoor 3D maps through classifying groups of points into floors, ceilings and various other objects. The ability to exploit semantic knowledge is limited when the number of objects becomes large, requiring an adequate way for structuring properties of and relationships among objects.

Another example of classifying indoor environment into semantic categories, [Shi et al. 2010] proposes a methodology for a robot where the classification task, using data collected from a laser range finder, is achieved by a machine learning approach employing logistic regression. Instead of gross categorization of locations as in the conventional approaches, this method shows the ability to classify parts of a single laser scan into different semantic labels.Semantic knowledge has been also used to support the building of semantic maps for autonomous robots from laser scans. For instance, the authors of [Goerke and Braun 2009] present a framework to build semantically annotated maps from laser range measurements of a mobile robot. The approach classifies an indoor environment to build an annotated grid map by using features that are extracted from the original laser range measurements. This allows calculating a class membership vector for the robot position. Another application using a mobile robot in classifying the different areas in indoor environments is also reported in [Mozos 2008]. The system uses semantic classes as information for representing the environment and extracting topological maps. The goal of this approach is to classify the position of the robot based on the current observations taken by the robot. In the current position, geometrical properties are encoded to be a set of features. They are then used to classify the scan into the corresponding

semantic class. The approach reduces the exploration and localization time of the robot.

In [Stueckler and Behnke 2011], knowledge such as floor, chairs, shelves and other semantic information have been used to support the detection and awareness of people in a service-robot's environment. The other information about the a priori likelihood that people are present at semantically distinct places is also used. Besides, the approach utilizes scene semantics to support robust detection and awareness of people in the robot's environment. The problem of detecting complex objects in the 3D scan of an indoor environment has been addressed in [Rusu et al. 2009] as to allow a personal robotic assistant to classify regions and estimate 3D geometrical features. The method employs a combination of semantic 3D object maps and triangulated surface maps. The system was designed to use either geometric mapping or learning to process large input datasets and object extraction. The concerned objects are kitchen appliances, cupboards, tables and drawers. These objects have been modeled accurately enough to be used in physics-based simulations where doors of 3D containers can be opened based on their hinge position. The result shows a map that comprises both the hierarchically classified objects and triangular meshes.

As far as the use of knowledge for object detection/scene annotation is concerned, some existing methods, such as [Wuenstel and Moratz 2004], only consider specific knowledge about each individual object and do not exploit the relationships – whether topological or semantic – that may exist between objects. This method however is restricted to basic objects but not limited to a special form. It goes without saying that inter-object relationships are important and can only facilitate object detection and improve the quality of the labeling. For instance, the combination of topological constraints with scene similarity has been proposed in [Posner et al. 2008] to support scene clustering. The method operates on a single matrix that expresses the pairwise similarity between all captured scenes. A concept of using sequence of algorithms is used and integrated with spatial constraints provided by the continuous motion of the vehicle. The problem of acquiring such relationships has been addressed in [Triebel et al. 2007a] in which relationships between classes of objects are modeled from training data sets. The work is proposed as annotation for different places and objects in 2D or 3D maps and shows how to choose the features representing the points in a map, and applying Associative Markov Network (AMN) following the concept of collective classification to classify sets of these features.

Some other approaches make use of hierarchical description of the objects' (or a scene's) attributes. In this respect, building facades are segmented in [Teboul et al. 2010] segmented using a derivation tree representing the procedural geometry, the connected grammar semantics and images. This approach proposed a dynamic way to perform a search through a perturbation model. [Ripperda and Brenner 2006] also extracted building facades using a structural description and used reversible-jump Monte Carlo Markov Chains to guide the application of derivation steps during the building of the tree. Another application of using knowledge is to infer the missing parts from detection. For example, [Pu and Vosselman 2009] reconstructed building facades from terrestrial laser scanning data. Knowledge about size, position, orientation and topology is used to recognize features (e.g. walls, doors and windows) as well as to hypothesize the occluded parts. In a similar work [Scholze et al. 2002], a model-based reconstruction method has been proposed. In this method, semantic knowledge is also used to infer missing parts of the roof and to adjust the overall roof topology. These approaches use knowledge to evaluate results from numerical processes, but do not integrate it into the processing as such.

2.5/ KNOWLEDGE-BASED APPROACHES

As discussed earlier, the use of semantic knowledge in evaluating results may compensate for missing data in traditional numerical processing methods. Semantics are only one possible kind of knowledge that can actually be employed to devise algorithms dealing with complex situations in a flexible and intelligent manner. In general, one would like to integrate human knowledge into processing. Such systems are capable of understanding the meaning of available sources (input data) to infer a proper strategy in processing. We review in the following the literature on knowledge-based technologies and techniques.

Data, information and knowledge

In order to understand the knowledge management as well as knowledge-based systems, it is important to have a working understanding of the differences between data, information and knowledge. In [Kahn and Adams 2000], the authors provide an overview of the terms:

Data is a collection of facts, unprocessed collection of details, with no purpose, value and meaning. In particular, the same data may be represented differently from one domain to another. For instance, data can be symbols which deliver a message, or raw numbers corresponding to sales, invoices, return, etc.

Information is data when processed to be useful, depend on the purpose of using information. Data is organized, summarized or analyzed in a different way. The authors gave an example about trend analysis of sales in which data provide information about a performance of a company.

Knowledge is an application that shows how to use data and information, when information is combined with context and experience. Knowledge provides implications and presents strategies on which to base decisions. Knowledge may be viewed from several perspectives: a state of mind, an object, a process, a condition [Alavi and Leidner 2001] of having access to information. For example in the forecasting context, the forecasting analysts use results of trend analysis (information) to draw inferences. From intuition and their experience from similar trend statistics from other product lines, the analysts are able to give an action plan. The more observations the analysts have, the more trend analysis is enhanced. Knowledge is the result of fusing information with practice.

The three key points of knowledge which we borrow from [Alavi and Leidner 2001] are as follows:

(1) A great deal of emphasis is given to understanding the difference among data, information, and knowledge and drawing implications from the difference.

(2) Knowledge is personalized. In order for an individual's or a group's knowledge to be useful for others, it must be expressed in such a manner it is interpretable by the receivers.

(3) Hoards of information are of little value; only that information which is actively processed in the mind of an individual through a process of reflection, enlightenment, or learning can be useful.

Knowledge management

When the amount of data, information and knowledge become large and complex, an efficient organization of knowledge is required. Such organization is critical for managing the storage resources as well as rendering access and inference time minimal. Knowledge management has been considered to identify and leverage collective knowledge and has become become an important tool in several firms allowing them to gain a competitive advantage.

The earlier knowledge management implementations focused on Information Communication Technology. It expended across all types of companies and organizations worldwide as mentioned in [Bechina and Ndlela 2009]. For example, an application of knowledge management in law firms [Gottschalk 2002] allows enhancing and abridging between implementing, sharing, distributing, creating and comprehending the knowledge of the organization. First the authors defined law firms in term of knowledge organizations. Then, knowledge management was presented in terms of the knowledge-based view of the firm. Finally, knowledge categories in law firms are mentioned. Another example, this time in [Alavi and Leidner 2001], lists out the appearance of knowledge management in stock firms and consulting companies in which semantic memories have been created by developing vast repositories of knowledge about customers, projects, competition, and the industries they serve. The recent interest in organizational knowledge has prompted the issue of managing the knowledge to the organization's benefit.

Also in a commercial domain, [Durand et al. 2007] describe several knowledge management processes. For instance, knowledge identification comprehends the attributes of the required knowledge. The knowledge acquisition process focuses on discovering the required knowledge such as buying, consulting, researching, developing and self-creating. Presenting information is carried out by the knowledge preparation process. Knowledge dissemination ensures the distribution of knowledge. Lastly, the knowledge maintenance process maintains a knowledge management system up-to-date. However, a knowledge management project implementation could be different from one application to another. For example, the authors of [Karadsheh et al. 2009] present a knowledge management process that allows saving time, efforts and avoids inaccuracies.

Knowledge-based systems

Knowledge management is applied to serve different particular purposes. These applications are knowledge-based systems. Such systems use knowledge-based techniques to support decision-making, learning and action. A review of knowledge-based systems has been reported in [O'keefe and Preece 1996]. The objective of a knowledge-based system is to replace or augment a decision making task. Its success is often dependent upon understanding of that task, its role in relation to other tasks, and its integration with other tasks (both manual and automated). A knowledge-based system is applied in plenty of domains such as accounting, finance, computer science, etc. There are five types of tasks that appear to be particularly successful and which we discuss in the following.

First, cumulative-hurdle decision making: This is where a number of decisions are made linearly, but the problem may be solvable without overcoming every decision making hurdle. A good example is loan approval where a knowledge-based system can handle the first hurdle — logical consistency of the loan application, basic credit worthiness, etc. When a "reject" or "grant" decision cannot be made, the application can be passed over to an expert for further review.

Second, advisory systems: they give simple advice to someone performing a task, such

as machine repair, collection of audit data, performing statistical experiments, etc. They are typically applicable to any well-defined task that requires the necessary expertise, and usable by a number of different users. They are beneficial when knowledge has to be distributed to professionals due to changes in the law, redesign of machinery, etc.

Third, heuristic systems: they produce solutions generated from mathematical models by using heuristics, but not in a reasonable amount of time or in a robust manner (e.g., infeasibility cannot be easily identified). As might be expected, they tend to be more quantitative than other knowledge-based systems, and often appear in the same domains as Operational Research models, for example, production scheduling.

Fourth, configuration systems: they take a requirement for a configured assembled product (such as a computer, or an air conditioner) and generate the parts needed to configure the product with associated assembly instructions. The major benefit of these systems is the ability to collapse the order processing cycle, such that an order can be configured and specified for manufacturing within hours, rather than days or weeks.

Fifth, critiquing systems: they are also known as expert critics. These produce critiques of a design or plan that has been produced by a user. They can either be activated by the user as required, or can run in the background, effectively ”looking over the shoulder”, monitoring the user’s actions and suggesting changes when user actions appear to be different from what the critic would do.

There has also been a growing interest in developing knowledge-based systems for various data processing tasks such as data segmentation and registration but also for scene understanding and interpretation. For instance, [Trinder and Wang 1998] have proposed a knowledge-based method which automatically extracts roads from aerial images. The description of roads includes radiometric, geometric properties and spatial relationships between road segments, all formulated as rules in PROLOG. The knowledge base stores structures of roads and relationships between them yielded from images. By using topological information of road networks, the method is able to predict missing road segments. However, the used semantic model is limited to one type of objects (roads). [Growe and Tonjes 1997] present a knowledge-based approach for the automatic registration of remotely sensed images. Knowledge is explicitly represented using semantic nets and rules. Prior knowledge about scene objects and a Geographic Information System (GIS) are used to select and match the best set of features. [Matsuyama 1987] proposes a method for automatic interpretation of remotely sensed images. The approach emphasizes the use of knowledge management and control structures in aerial image understanding systems: a blackboard model for integrating diverse object detection modules, a symbolic model representation for 3D object recognition, and integration of bottom-up and top-down analyses. Two kinds of knowledge are considered in their expert system: knowledge about objects and knowledge about analysis tools (e.g. image processing techniques).

[Rost and Muenkel 1998] proposed a knowledge-based system that is able to automatically adapt image processing algorithms to changes in the environment. The method uses expert knowledge that is explicitly formulated by rules. Depending on a given task, the system selects a sequence of relevant image processing tools and adjusts their parameters to obtain results with some predefined quality goals. Results on object contour detection, carried out in various conditions, show the benefit of taking expert knowledge into account for adjusting the parameters of various image processing operators.

Focusing on the application in Computer Vision and Robotics, [Okada et al. 2007]

presents an object recognition subsystem of knowledge-based vision-guided humanoid robot system. The approach introduces a visual object recognition system based on multi-cue integration and particle filter based stochastic approach. The system is able to be utilized for both navigation and manipulation tasks by using movable or fixed knowledge of the object. The authors present a knowledge-centered integration of vision and motion subsystems. This approach enables the subsystems to perform effectively by communicating with each other through shared knowledge. However, limitations of the system are:

1) knowledge needs to be modeled manually. Development of manipulation and visual knowledge acquisition behavior is required.

2) The system does not recognize other robots or humans. Human or robot activities recognition and integration with an object recognition system are required.

3) The object recognition subsystem and the motion planning subsystem are tightly connected and integrated by sharing the same object and environment knowledge. This feature makes possible to automatically generate visually-guided behaviors.

Since the use of knowledge within processing is also useful, other research has focused more on knowledge management in terms of computation. For example, [Maillot and Thonnat 2008] used a visual concept ontology composed of visible features (such as spatial relations, color and texture) to recognize objects through matching among numerical features and visual concepts. [Durand et al. 2007] proposed a recognition method based on an ontology which has been developed by experts of the domain; the authors have also developed a matching process between objects and the concepts of the ontology to provide objects with a semantic meaning. However, knowledge in these approaches has not been fully exploited: other capabilities, such as processing guidance, have not been explored.

This previous research efforts show that there have been various attempts to devise a more robust and efficient analysis of point clouds. It emerges that such analysis requires structured processing going from most to less prominent characteristic features and bridging between the objects and their expected geometry. Simple models are efficient and robust but have limitations for more complex objects. Statistical methods are able to handle more complexity. They, however, also require large training efforts and are difficult to transfer. Knowledge-based methods seem to have the potential to manage complex scenarios. Successful work uses geometric and/or topological relations of objects for their identification, or attempts to map the structure of a scene into a semantic framework. Other works introduce knowledge into the processing and allow the use of various characteristics of the objects in order to improve their detection.

Various kinds of knowledge-based approaches appear in application development, documenting an increasing interest for semantic approaches. This expresses a certain expectation about the role of semantics in future solutions. What is still missing is an overall approach for knowledge integration, which would guide the numerical processing, the evaluation, and classification of the found objects.

3

BACKGROUND

3.1/ SEMANTIC KNOWLEDGE

3.1.1/ KNOWLEDGE-BASED SYSTEMS

Information and communication technologies play a vital role in many domains. The presence of information technologies in strategic fields such as economics, finance, education, health care, security, etc. clearly shows their importance. The proliferation of such technologies has given rise to numerous challenges as to reaching high levels of precision and reliability.

3.1.1.1/ ARTIFICIAL INTELLIGENCE PERSPECTIVE

The field of Artificial Intelligence research has been founded in 1950s and has since provided key solutions for many novel applications. It is a branch of science that copes with machines to solve complex problems and acts in a manner that inclines to call intelligent. The goal of Artificial Intelligence is to develop systems that exhibit an intelligent behavior somewhat like humans.

The main concern in Artificial Intelligence is to devise concepts, methods of symbolic inference and reasoning. The key challenge lies in the machine representation of the knowledge in such a way it can be used to make inferences. In general, Artificial Intelligence attempts to borrow characteristics from human intelligence and simulating them as algorithms in a computer. It is usually associated with Computer Science, and yet it has many important connections to other fields such as Psychology, Philosophy, Mathematics, Biology and others. Artificial Intelligence techniques come with their share of advantages and drawbacks [Shermarc 2002]

Positive outcomes

+ The biggest advantage of Artificial Intelligence is tireless performance of tasks. Unlike human who needs to take time and pauses to think, machines can get a specific task done without having a break.

+ The positive changes in factories, such as increasing production and efficiency,

indirectly lowering costs and decreasing errors can result in applying artificial intelligence.

+ Artificially intelligent robots potentially replace human in dangerous tasks (potential accidents and unsafe conditions), such as radioactive elements, confined space, little oxygen to breathe, etc. This replacement reduces the risk and unwarranted deaths.

+ Artificially intelligent robots are increasingly concerned to replace in menial tasks. On the one hand, this is because of the demands in the society (public service machines, elderly and disabled persons need help), and, on the other hand, because the systems based on Artificial Intelligence are likely to be more accurate and increase the level of trust in taking certain decisions.

Negative outcomes

- The use of artificial intelligence in everyday tasks more or less produces laziness on the part of humans. However, because humans have an extraordinary ability to think, analyze, and use judgment, Artificial Intelligence is unlikely not replace humans completely.

- Artificial Intelligence is effortless performance of a job, and thus any artificial intelligence-based system contains the risk of a breakdown or a loss of data. In certain cases, a machine can fail to keep data within its memory due to the malfunction of certain components and next process will fail as well. This error can also happen with humans when they forget collecting and saving data. However, human can change flexibly to adapt with new situations a machine only obeys a given program.

Glancing over the achievements accomplished by Artificial Intelligence in different fields it is to be noted that the contributions of Artificial Intelligence are remarkable [Bit-tech 2012]. For instance, banking systems employ algorithms that are capable of recognizing unusual patterns in customer spending and detect credit card fraud. In security systems, facial recognition is often installed at airports. Other systems are able to identify vehicles based on recognizing their registration number right from the number plate. Both facial and number recognition are algorithms that employ Artificial Intelligence techniques. Web search engines, which are used by millions of people every day are also based on Artificial Intelligence. Speech and/or voice recognition are now integrated into operation systems. The field of Artificial Intelligence has brought solutions in many fields and made computer programs exhibit an intelligent behavior.

Taking human cognition into account is nowadays a key research direction in the field of Artificial Intelligence. Lots of efforts are focusing on capturing human knowledge and embedding such knowledge explicitly into systems. The goal is to provide a computer with the same knowledge a human may have in some domain.

3.1.1.2/ DATA, INFORMATION AND KNOWLEDGE

Languages are used to communicate and express meaning. In each context, one aims to talk about a specific topic. Languages contain data and information. However, the underlying knowledge often takes on a variety of meanings. Answering questions such as ”What is knowledge?” and ”What is information?” has been a major problem of philosophers and scientists since ancient times. This section is to contribute to an overview regarding the terms data, information and knowledge within the Computer Science community in general.

Many studies claim that data, information and knowledge are part of a sequential order. Data are the raw material for information, and information is the raw material for knowledge. If this hypothesis is approved, then Information Science should explore data and information, but not knowledge, which is an entity of a higher order. However, information science does not seem to explore knowledge since it includes knowledge organization and knowledge management, which can be confusing [Zins 2007]. From the previous research [Davenport and Laurence 2000], [Trainmor 2013], we collect and distinguish between the meanings of the three fundamental concepts of data, information and knowledge as follows:

Data

Data are raw facts or figures about an event. They have no meaning on their own. Data can be any numerical quantities (cost, time, speed, and capacity), text, symbols or attributes derived from observation. These materials do not provide judgment or interpretation, and they are not organized in any way, but raw material of decision making may include data.

Information

Data, when organized with relevance and for a certain purpose, becomes information. Information is a collection of data, or associated interpretations to describe a particular meaning. As in [Davenport and Laurence 2000], there are five main processes to convert data to information:

Contextualization: data is collected following a purpose or reason based on what we already know.

Categorization: we process data as to assign each into a proper type or category.

Correction: is a process to remove noise (or errors) from data.

Condensation: data is summarized in a more concise form.

Calculation: we aggregate and analyze data to obtain useful information.

Knowledge

Knowledge is the combination of framed experience, contextual information and expert insight, all of which are mixed to become a framework that is able to evaluate and incorporate new experiences and information. In each specific field, knowledge appears not only in documents, books but also in routine activities and processes. Knowledge is derived based on collecting information in an appropriate way that is carried out by humans.

Basically, from the activities of human or a group of individuals such as when people exchange information or have conversations, we obtain knowledge about a specific domain [Davenport and Laurence 2000].

There are two primary kinds of knowledge: explicit knowledge and tacit knowledge are mainly considered.

Explicit knowledge is knowledge expressed or found in formal languages, statements or certain expression types. It is easy to exchange from person to person, can be stored in data bases or processed by computers.

Tacit knowledge is knowledge we can find from experience or insights of human. This kind of knowledge usually refers to an implicit comprehension and is not intuitively expressed through language. However, to describe tacit knowledge in forms that people can understand, we can represent this knowledge through languages such as by words, numbers, symbols or other types of representation.

Wisdom

Wisdom is the ability to make correct judgments and take appropriate decisions on the bases of previous knowledge, experience and insight. Humans gained experience and accumulated knowledge through activities over time. From accumulated knowledge and experience, humans can create sensible judgments and make wise decisions in a certain situation or event. Thus an individual with wisdom can reach a goal by applying appropriate knowledge into the way he processes [OTEC 2007].

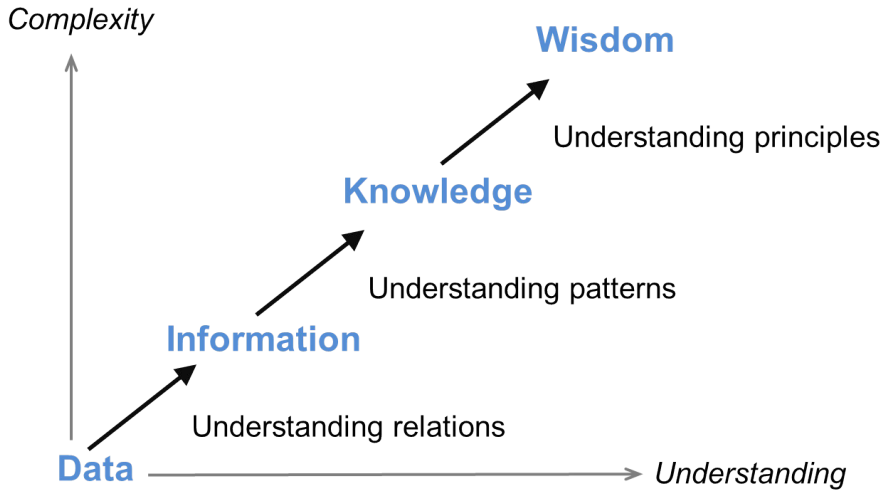


Figure 3.1: A progress from data becoming to wisdom

Examples:

Data represents a fact or a statement of event which is independent from other things.

e.g.: Red

Information contains a meaning that can be a relationship between events, or possibly cause and effect.

e.g.: A traffic light has turned red.

Knowledge represents a pattern that connects and generally provides a high level of predictability as to what is described or what will happen next.

e.g.: At a cross where my car is approaching, a traffic light has turned red.

Wisdom embodies more of an understanding of fundamental principles embodied within the knowledge that are essentially the basis for the knowledge being what it is. Wisdom is essentially systemic.

e.g.: I should stop the car.

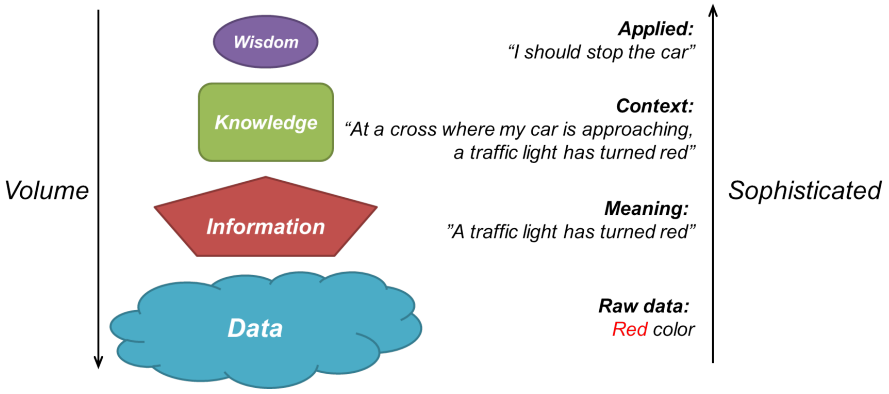


Figure 3.2: Illustration of the example about the differences between concepts of data, information, knowledge and wisdom in a context

3.1.1.3/ EXPERT SYSTEM OR KNOWLEDGE-BASED SYSTEM

Knowledge-based systems are computer programs belonging to the branch of Artificial Intelligence. A knowledge-based system is an interactive computer-based decision tool that uses facts and heuristics to solve difficult decision making problems based on knowledge acquired from various sources. Such systems potentially solve problems such as (1) understanding the behavior of data or events, (2) learning from experience and (3) decision making. Traditionally, computers use computational algorithms to solve problems. With a knowledge-based system, the idea is to improve conventional methods by taking human knowledge and making it interact with a program to solve complex problems. A knowledge base is modeled from different sources such textbooks, documents, data, expert or non-expert knowledge. In a knowledge-based system, the knowledge base does not cover the entire knowledge from many fields but rather contains knowledge about a specific domain (any part of the world) or some degree of expertise in the problem of interest. The task of a knowledge-based system therefore is to solve issues within the range of a domain. For example, knowledge-based systems are used in medical diagnostic applications, financial planning decisions, monitor real time systems or educational aids.

Earlier, knowledge-based systems were known as expert systems. An expert system

is an Artificial Intelligence application in which a computer program acts intelligently by using the encoded knowledge of experts in field. Edward Albert Feigenbaum is widely known as the father of expert systems. The first expert system called DENDRAL, was developed in the early 70's at Stanford University and was applied in the field of chemistr. Later, the term "knowledge-based system" has been used more often than "expert system". However, both "knowledge-based system" and "expert system" are used synonymously to refer to such systems. While knowledge-based systems can employ knowledge from various sources, including non-expert and expert, expert systems often use knowledge created by a person or a group of people with special expertise in a specific area. When solving particular tasks, either knowledge-based systems or expert systems capture concepts and act as humans would do, for instance for aiding customers in complex regulations, for selecting products or diagnosing equipment problems, etc.

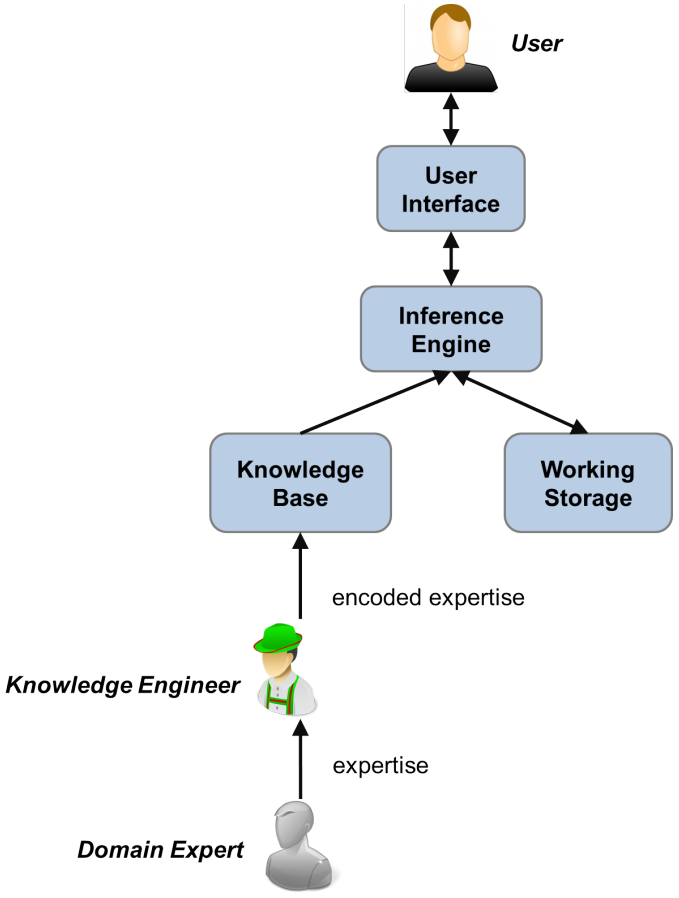


Figure 3.3: Intervention and major components in an expert system

Expert systems need persons/practitioners to model knowledge: they are knowledge engineers. Before modeling knowledge, knowledge is acquired from different related sources in the domain of interest. Knowledge acquisition includes the collection, analysis and validation of knowledge. Note that knowledge acquisition requires time. As a single expert may not know everything, this work sometimes requires several people to be completed. The reason is that even one might in general be interested in a specific domain, but there are still vast amounts of knowledge in this domain. The task of integrating knowledge into computer systems is known as knowledge engineering.

Knowledge engineers have the responsibility to make sure that the required knowledge

is sufficient for an expert system to function properly. Therefore, knowledge engineers play an important role in choosing the appropriate knowledge to feed into the knowledge base. For example, in an online shopping platform, knowledge engineers collect sources of related domain, such as properties of good, customer's information, payment methods, availability of goods in the store, etc. This knowledge is represented in a certain form and stored in the knowledge base. The knowledge base may also encode expertise in a domain of interest and used in the expert system. The formalization of knowledge is referred to as knowledge management [Chakraborty 2010].

User interface

The human-computer interface in expert systems allows the user to communicate with the system. The interface allows the user to enter information, to query and receive advice from the system. In reality, users often communicate with experts for obtaining advices to solve a problem in the range of the expert's knowledge. Expert systems are designed based on the same idea, i.e. the user interface in an expert system allows users and experts to communicate easily. The main purpose of a user interface is helping users easily communicate with the application. Interfaces could be menus, windows or advanced tools which require graphic support.

Working memory

A working memory in an expert system is typically used to receive a set of facts, information about a particular domain that is specific to the problem being solved. The actual data presented to the working memory depends on the type of application. Data may consist of the set of conditions leading to the problem. Working memory contains the facts from both supplement from the users and reasoning done by the expert system itself.

Knowledge base

Knowledge base is a set of rules which presents the knowledge about the domain of interest. Knowledge from expert is usually made up of heuristics or factual knowledge. Heuristic knowledge is knowledge gained after experiences. Factual knowledge includes information from textbooks, literatures, shared documents, common human knowledge, etc. Heuristic knowledge and the factual one consist of large information which is not structured in the way computers can understand. Transferring knowledge from the expert into a program in computer requires an encoding process that preserves the meaning of what the expert wants to deliver to the expert system. The representation of the expertise is often in forms of "if-then" rules, semantic networks or frames, they are the representative forms to store information in a logical and natural way.

"If-then" rules

"If-then" rules are often used to state "cause and consequence" sentences. In a specific domain, experts in this field think along the same way:

Condition → Action

Or

Situation → Conclusion

Such usual statements can be transformed into a kind of knowledge encoding such as a

"if-then" form:

If x1, x2, x3... Then y1, y2, y3...

Where xi is a condition or a situation and yi is an action or a conclusion.

Example:

"If it is raining then we will stay at home."

Condition is "it is raining" and action is "stay at home".

"If the humidity is high and temperature is low then it will probably snow"

Situations are "humidity is high" and "temperature is low", conclusion is "it will probably snow".

Semantic networks

Objects or events in the real world usually have relationships which reflect the mutual influences between objects. Representation in term of relationship between objects is known as a semantic network. Semantic networks are often represented as graphs whose nodes are objects. The links between nodes in the graph represent for relationships between objects. The common form of a semantic network is a graph that contains links between nodes using "is-a" and "has" relationships between objects. The phrase "is-a" is used when we present objects and classes being related to each other by a class relationship, while "has" is for representing one object belongs to or is a member of the other object.

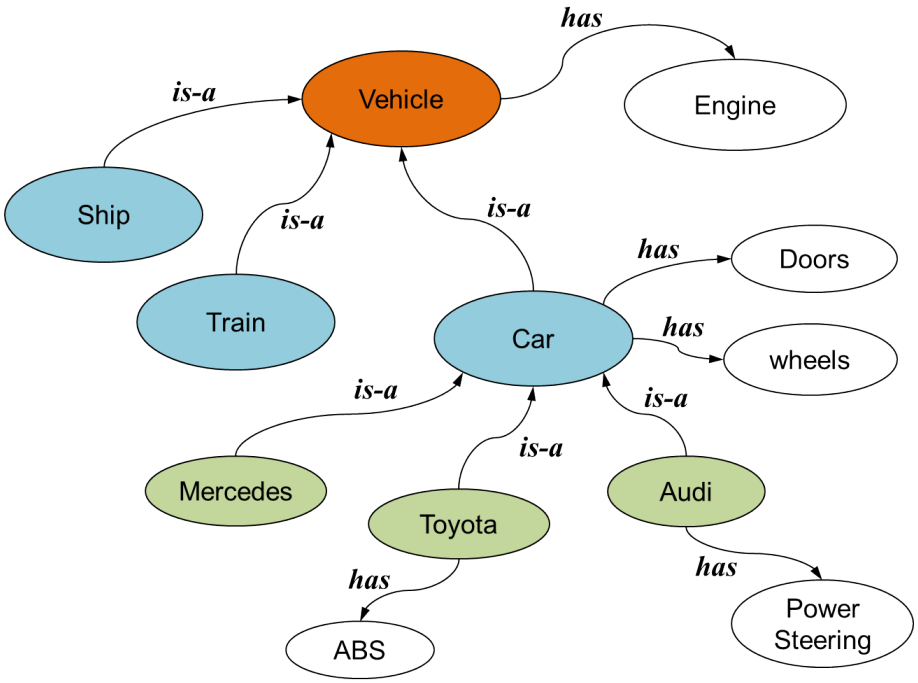


Figure 3.4: Relationships between elements in "vehicles" represented by a semantic network

Frames

A frame is a type record structure which contains concepts (or objects), situations, at-

Frame	Car
Inheritance slot	Is-a
Value	Vehicle
Attribute slot	Engine
Value	1
Attribute slot	Cylinders
Value	4
Value	6
Attribute slot	Doors
Value	2
Value	4

Table 3.1: A frame and its slots filled with data types

tributes of concepts, their relationships, and procedures to handle relationships and attribute values. A frame-based representation has frames and slots. A separate frame usually contains a concept and the attributes of concepts, the relationships between them. The procedures are allotted to slots in a frame. The content of a slot are data types (e.g. symbols, strings, numbers. . .), functions or procedures, etc. A single frame is not much useful. Frame-based representation usually have frames connected to each other through a similar kind of inheritance as that provided by a semantic network.

Inference engine

A knowledge base alone does not make an expert system become intelligent. An expert system usually has an important facility which is used to navigate and manipulate knowledge from the knowledge base to infer certain conclusions. This facility is an inference engine which is usually set up to simulate the reasoning that humans use to draw conclusions. The reference engine of an expert system makes use of the knowledge base to draw such conclusions in some given situations. Based on the given facts, the inference engine determines the set of rules that should be considered and matched to the current goal of the system [Nikolopoulos 1997], [Patterson 1990], [Rajeev 1996].

Two primary methods, namely forward chaining and backward chaining, are used for making inferences from the knowledge base. Forward chaining is a data-driven method. The system starts with the initial facts in the working memory, and keeps using the rules to draw new conclusions or take certain actions based on those facts. Backward chaining is a goal-driven method. The system starts with some hypothesis or goal and then keeps looking for rules that would allow confirming that hypothesis and taking certain actions.

Expert system characteristics

- The highest level of expertise: The most important and useful characteristics of an expert system are efficiency, accuracy and imaginative problem solving.

- Interactivity: Expert systems are defined as computer applications which embody human expertise in aiding the decision making process. Such systems not only perform the defined functions and given tasks but also make it possible to respond to questions from users as well as request for clarifications. The aim of expert systems is not replacing an expert to solve a problem but rather generally require

a human–computer interaction which draws from supporting knowledge from both the machine and the human side. An expert system must interact in a reasonable time with the user, and this time must be less than the time taken by an expert to solve the same problem.

- Knowledge is stored and sifted: Like human knowledge, knowledge in expert systems is stored in working memory. Typically this knowledge is obtained from a human expert through experience in a domain over a period of time. Depending on the complexity of the domain, knowledge engineering could take a few days to a few years. Knowledge is not fixed but also expandable due as processes may be updated over time. Thus, knowledge in expert systems is accumulated. Knowledge engineers carry out checks on the completeness and correctness of the knowledge presented to the system.

- Making logical inference: A knowledge base contains sets of rules which represent the knowledge about the domain. Unless these rules have means of exploiting the knowledge that is stored in the working memory, they are useless. The reasoning mechanisms make use of an inference engine to process rules in the knowledge base.

- Domain-specific expertise: An expert system cannot have a knowledge base covering all possible fields. Only some parts of the world are captured within the system. A particular system only supplies a narrow area of specialization. Most experts are knowledgeable and skilful in their own domain only. Therefore expert systems are made to focus on a specific domain without mixing up the knowledge of two experts from different domains.

- Making mistakes - Uncertainty: Human experts transfer their point of view in the real world to an expert system. The knowledge from exerts can be imperfect which causes uncertainty. The knowledge may therefore contain errors in the facts, in the rules or in the input during the process of acquiring of knowledge. This may yield an incorrect output from the side of the expert system [Chakraborty 2010].

3.1.2/ KNOWLEDGE ACQUISITION

Human beings often learn knowledge about a particular field from the related facts and learn best from experience over time. Through the basic senses (touch, taste, smell, sight, and hearing), humans perceive information from outside and transfer to the brain. An expert system also needs knowledge in the domain of interest to be able to operate. Unlike human beings, the expert system is not able to perceive knowledge through senses. It, however, needs support from both human experts to extract knowledge from different sources and knowledge engineers to pass the encoded knowledge (in the form of the knowledge representation used) from experts to the system. This process is known as knowledge acquisition or knowledge engineering. Knowledge acquisition includes collecting, modeling, analyzing and validating knowledge. Knowledge about a

specific domain can be of any form such as symbols, text, sound, pictures, etc., which are often raw data. Knowledge can be acquired in various ways:

Interviews

The classical method of knowledge acquisition is the interview. Interview is a simple method to exchange routine information between people. However, interviewing experts to acquire their knowledge in a specific domain must be structured as opposed to randomly selected questions that do not allow capturing and preserving the entire information. A structured interview is carried out through a formal method that guides both knowledge engineer and expert following a given scenario. Lists of questions related to the domain of interest as well as lists of answers are created. A question can have a few possible answers and the expert can select one of them (or a multiple selection). Another task consists in modeling the rules in the system. This work should be done based on the expertise of experts. Knowledge acquisition through interviews can be used for any domain, does not require much time from the expert. However, the difficulties are sometimes that the experts do not want to follow the given questions in the interview which requires the knowledge engineer to have a certain prior knowledge to be able manage such situations during an interview.

Protocol analysis techniques

The intent is to capture and report the activities of experts to model knowledge. The knowledge engineers often use transcripts of interviews or text-based information to record various types of knowledge such as target, decision, relationships, etc. The knowledge engineer can interrupt the expert at critical points to ask questions. This typically occurs when the knowledge engineers needs to know why the expert performed a particular action. Such interruptions might distract the expert. This method is usually time-consuming.

Observation techniques

This technique is another way of generating protocols. The knowledge engineers observe the expert performing a task, take notes or use recording equipment. This method is simple but may be time consuming.

Diagram-based techniques

Knowledge engineers employ network diagrams, such as concept maps, state transition networks and process maps, to capture the ”what, how, when, who and why” of tasks and events. The representation of acquired knowledge in a network format can be easier to visualize and makes validation very efficient. This technique is particularly useful when exploiting knowledge from a complex domain [Chakraborty 2010].

Knowledge acquisition is considered as a difficult process in the development of expert systems since it requires several tasks from human experts. Such tasks consist in observing the facts, perceiving the knowledge from the facts. This is followed by understanding, selecting and then transferring knowledge to the system. Knowledge is obtained from the experts as they perform a certain task and archives experiences. Experts, who are

knowledgeable in their specific domain, have vast amounts of knowledge of which a great deal may be tacit and which is difficult to describer. However, one expert does not know everything but mostly know deeply in a specific field. The progress of knowledge acquisition depends on the complexity of the domain and the availability of experts (as a fact the experts are often busy), knowledge engineering could take anywhere from a few days to a few years to complete the knowledge base for an expert system.

3.1.3/ KNOWLEDGE REPRESENTATION

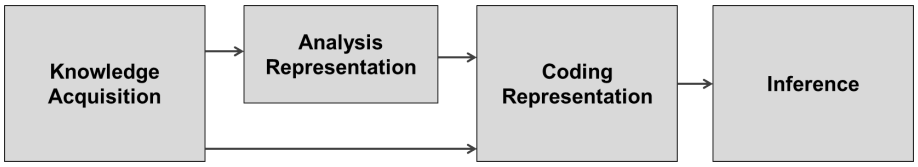


Figure 3.5: Progress of knowledge representation

Computer programs are not capable of understanding the knowledge captured by the knowledge engineers from experts. Knowledge representation is performed to encode the acquired knowledge into forms that are useable for computer programs. A knowledge representation appears in different forms, the most popular ones are semantic networks, rules and logical presentations. Semantic networks are graph representations consisting of categories of objects as well as the relationships between objects. Knowledge can also be represented by rules which usually appear as ”if-then” constructs. Logical representations are formalized through semantic networks and rules to give them a precise semantics. These forms are introduced in the following sections [Grimm et al. 2007].

3.1.3.1/ SEMANTIC NETWORKS

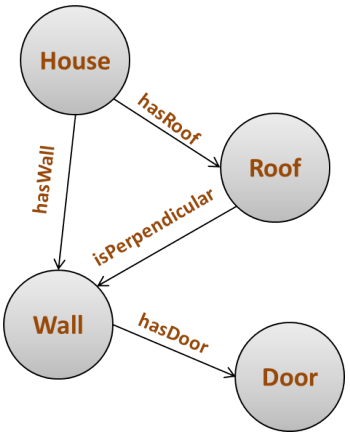


Figure 3.6: Example of a semantic network

Knowledge acquired from experts is often represented through natural language, image, voice, etc. These representations are familiar to humans but not to computers. Semantic networks provide a way of interpreting the meaning of knowledge by a visual graph. Semantic networks formalisms express the taxonomic structure of categories of

concepts and relationships between them. A semantic network is a graph which consists of nodes and arcs. Nodes represent concepts while arcs denote relations between concepts. Statements about a domain of interest are represented through nodes and arcs in the graph.

In a semantic network, concepts (nodes) are often nouns such as text and the relations between concepts (arcs) are verbal phrases. The paths connecting concepts and their relations represent statements in the domain of interest. Concepts are usually objects in a relevant domain. However, relations can be generic (e.g. action, property, etc.). Relations can be either a specific property/action or a general one which used in other domains as well. For example: ”is a”, ”is a kind of”, ”consist of”, etc are general relations.

A few properties (e.g. negation, disjunction, and general non-taxonomic knowledge) are not easy to express in semantic networks. To express those properties, we can use complementary predicates. Data values, such as numbers and strings are also not represented in a semantic network.

3.1.3.2/ RULES

Rules are another form of expressing knowledge and allow expressing the notion of consequence. Typically, the rules are often of the form “if-then” constructs that allow presenting different kinds of complex statements. The “if-then” rule statements are used to formulate the conditional statements that comprise fuzzy logic. The If-part is referred to as premise and the then-part is referred to as conclusion.

For example, a single “if-then” rule can be:

$$\text{if } x \text{ is A then } y \text{ is B}$$

where A and B are values in the respective ranges of x and y.

The example above is a kind of simple single rule. Complex rule can contain two premises and one conclusion or can be a combination of many single rules. The phrases “if-then” are understandable for humans but not suitable for computation.

For example:

If a building has a roof and has walls then it is a house.

Such phrases are formalized to use predicates and variables over objects of the domain of interest. A typical style of rule after being formalized looks as follows.

$$house(?b) : -building(?b) \wedge hasRoof(?b, ?r) \wedge hasWall(?b, ?w)$$

In logic programming systems, a rule is often described as a conclusion first and then a symbol “:-“ used to connect a premise afterwards. Variables start with the symbol ? and take as their values the constants that occur in facts. Multi-premises are combined by a intersecting symbol “$\wedge$”. A system that uses the rules to represent the statements in the knowledge of a specific domain is known as “rule-based system”. The complexity of a “rule-based system” reflects through the number of rules used in the system and the number of premises in each rule.

3.1.3.3/ LOGICAL REPRESENTATION

Humans speak in different languages and this sometimes leads to some sorts of misunderstanding in conversations. The reason mostly comes from linguistic representation. To reduce this, we must understand the rules of the language (that we are using to present information in the conversation) such as the syntax and the semantics. Each language has a particular syntax in terms of using symbols, words and construction of sentence. Semantics of a language is what the sentence means. Logic representation consists in using a language and agreeing upon its rules when representing information [Colton 2010].

The term “predicate logic” is usually used in logical presentation. Predicate logic is defined as the general symbolic formal systems such as first-order logic, second-order logic, etc. Two common quantifiers in predicate logic are the existential ∃ (”there exists”) and universal ∀ (”for all”) quantifiers. Those differentiate predicate logic from other systems.

The popular logical representations are Propositional logic and First-order predicate logic:

Propositional logic

Propositional logic is a restrictive logic which can express a possible condition of the world as propositions or statements in a sentence. Propositional logic can have a complicated proposition (or statements) by combining simpler propositions (or statements). To join two propositions, we use connectives “and” ($\wedge$) or “or” ($\vee$). The complex statement formed by “and” is true if and only if both the component statements are true. The complex statement formed by “or” is true if one of the component statements is true.

For example:

When we say: “If a building has walls and it has also a roof then the building is a house.”

The statement can be expressed in a form of logic description as:

A building has walls $\wedge$ *A building has a roof* $\rightarrow$ *The building is a house*

The meaning of the sentence is when we know the propositions (a building has wall and a building has a roof) are true then we can assign truth values to a sentence.

First-order predicate logic

First-order predicate logic is built on propositional logic and allows us to employ not only the connectives (“and” or “or”) to join the statements as in propositional logic but also constants, variables, functions, predicates, quantifiers (existential ∃ and universal ∀). First-order predicate logic allows us to express: “All buildings having roofs and walls are houses” as:

$\exists$ x (Building (x) $\wedge$ hasWalls (x) $\wedge$ hasRoof (x) $\rightarrow$ house(x)).

3.1.4/ ONTOLOGY IN INFORMATION SYSTEMS

(Ref: T. R. Gruber. A translation approach to portable ontologies. Knowledge Acquisition, 5(2):199-220, 1993) The word “ontology” originally comes from philosophy, and the meaning of “ontology” refers to the subject of existence and is about knowledge as well as

knowing. In the context of knowledge sharing in information systems, an ontology refers to a specification of a conceptualization. An ontology is a description of the concepts and relationships that can exist for an agent or a community of agents. In the context of Artificial Intelligence systems, knowledge in a knowledge-based system is often represented by a set of representational terms which are the concepts and their relationships among concepts. Ontology used in a knowledge-based system contains entities in the universe (e.g., classes, relations, functions). The name of entity is human-readable text describing what the names mean, and formal axioms that constrain the interpretation and well-formed uses of these terms.

(Ref:" Knowledge Representation and Ontologies Logic.pdf" or Book: "Scientific Data Mining and Knowledge Discovery") An ontology has several characteristics like formality, explicitness, being shared, conceptuality and domain-specificity which are introduced in the following

- Formality: Representing knowledge through an ontology must preserve the original information of extracted knowledge. Besides, the way of interpreting knowledge in an ontology must also be well-defined so that machine can access and process.

- Explicitness: If an ontology does not clearly and logically represent notions, the machine will not be able to understand entirely the knowledge that human want to deliver. Explicitness is an important characteristic that avoids ambiguity when transferring information from knowledge to machine.

- Being shared: To have the same conceptualization in a large community is difficult. However, it is easier to have an agreement on a conceptualization in a particular group of people. An ontology is always limited to a particular community and its construction is associated with a social process of reaching consensus.

- Conceptuality: In ontology, knowledge is interpreted in a conceptual way that the concepts and their relations are presented by symbols. An ontology describes a conceptualization in general terms and does not only capture a particular state of affairs but attempts to cover as many situations as possible.

- Domain specificity: An expert cannot know everything in the world but focuses only on a specific domain. Knowledge, acquired from the fact by experts, is also limited in a particular domain of interest. An ontology often covers the details of a domain rather than a broad range of related topics. However, it is possible to separate a domain of interest to a narrower topic that can be represented in a single ontology.

Conceptual modeling with ontology is similar to designing the relationships in object oriented software. However, structure of ontology language is more advanced than such object oriented language. Ontology provides rich formal semantic knowledge in a specific domain. Moreover, an ontology can be used for different purposes such as data storage and reasoning about domain knowledge. To make the means of domain knowledge available to machine, using ontology language to represent knowledge is one technique that can conceptualize the concepts in a machine-interpretable way. Detail about the ontology

language is introduced in the following section.

3.1.5/ Ontology Languages

In the context of the Semantic Web, ontology languages play a particularly important role. The general concept of a Semantic Web is to annotate web content by machine-interpretable meta-data that allow computers to process this content on a semantic level. There are several characteristics from Semantic Web that also affect the use of ontologies for semantic annotation such as the distributedness of knowledge [Grimm et al. 2007].

3.1.5.1/ Web Ontology Language (OWL)

World Wide Web Consortium (W3C) standardization efforts have produced the Web Ontology Language (OWL) family of languages for describing ontologies in the Semantic Web. OWL is a language for semantic annotation of web content and is accepted within the Semantic Web community. OWL has different degrees of expressiveness such as: OWL-Lite, OWL-DL and OWL-Full. The OWL-Full is the most expensive one while OWL-Lite is the least expensive one.

OWL-Full: There are no limitation of how and where to use the language constructs. OWL-Full is designed to preserve some compatibility with Resource Description Framework (RDF) Schema [semantic web 2004] and is undecidable.

OWL-DL: A limited version of OWL-Full, OWL-DL has certain restrictions on how and where the language constructs can be used to guarantee decidability.

OWL-Lite: It is a subset of the OWL-Full and has a few limitations such as classes can only be defined in terms of named super-classes and only certain kinds of restrictions can be used.

Knowledge representation formalisms in the context of the Semantic Web nowadays often employs both description logic style ontologies and logic programming style rules to be interoperable on a semantic level. One attempt is the Semantic Web Rule Language (SWLR) that extends the set of OWL axioms to include Horn-like rules interpreted under first-order semantics.

The OWL abstract syntax presents an ontology as a sequence of annotations, axioms and facts. OWL allows to present content (to knowledge engineer) in a human readable text format or in more scientific context as using description logic formulas. The basic elements of an ontology are concepts, relations and instances which are also called classes, properties and individuals, respectively. In description logic, these three terms correspond to concepts, roles and individuals. The Web Ontology Language – Description Logic (OWL-DL) is constructed by two separate parts classes and individuals.

The components of OWL [semantic web 2007]:

OWL Classes

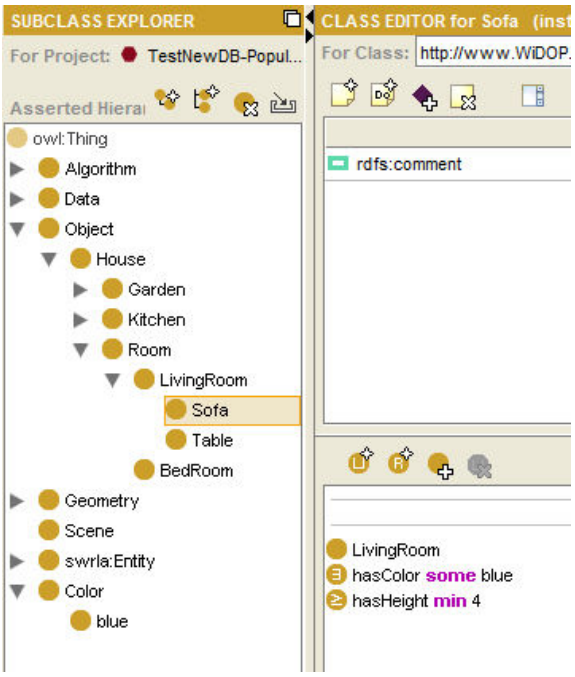


Figure 3.7: Classes and properties in an OWL

Classes are the basic building blocks of an OWL ontology. OWL supports six main ways of describing classes such as: Named classes, Intersection classes, Union classes, Complement classes, Enumerated classes. Among them, Named class is the simplest one.

- Intersection classes are formed by using two or more than two classes with the intersection (AND) operator.

- Union classes are formed by combining two or more than two classes with the union (OR) operator.

- Complement classes is specified by negating another class. Complement class contains the individuals which are not in the negated class.

- Enumeration class is specified by explicitly listing the individual that are members of the enumeration class. The members of the enumeration class are listed inside curly brackets.

Restrictions

Restrictions in an OWL provide a way of specifying local domain and range constraints. They describe a class of individuals based on the type and possibly number of relationships that they participate in. OWL Restrictions can be classified into three major categories:

- Quantifier restrictions:

- Existential means “some values from” or “at least one”. A class of individuals that have at least one kind of relationship along with a specified property of an individual that is a member of a specified class.

- Universal means “all values from” or “only”. For a given property, all the individuals have to be members of a specified class.

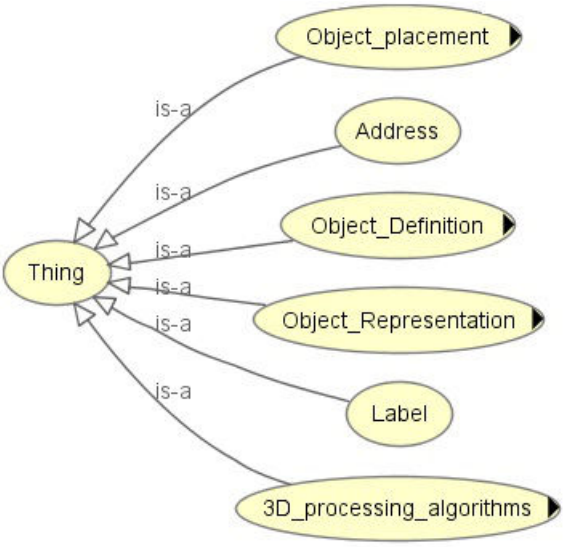


Figure 3.8: Example of relations between individuals in an OWL

- Has value restriction: used to specify that a class of individuals that participate in a specified relationship with a specific individual.

- Cardinality restrictions: For example: Min, Max, Equal. . . Cardinality restriction is the number of relationships that a class of individuals participates in.

Property

There are two major categories of properties:

- Object property: that links individuals to individuals

- Data property: that links individuals to data-type values (integer, string, float, etc.)

Property in OWL has some characteristics such as Functional (the property takes only one value), Inverse functional (the inverse of the property is functional). Furthermore, there are Symmetric and Transitive characteristics. A Symmetric property refers to a symmetric relationship, for example: if A is linked to B then B can be linked to A. A transitive property is an object property that defines a transitive relationship, for example: A is related to an element B, and B is in turn related to an element C, then A is also related to C.

OWL is the latest standard in ontology languages. It has a rich set of constructs and can perform reasoning over ontologies [Protege].

3.1.5.2/ The Semantic Web Rule Language (SWRL)

SWRL is an expressive OWL-based rule language which allows users to express the terms of OWL concepts in rules. SWRL provides more powerful deductive reasoning capabilities than OWL alone. Semantically, SWRL is built on the same description logic foundation as OWL and provides similar strong formal guarantees when performing inference. A SWRL rule consists of an antecedent part and a consequent part: the two parts include positive conjunctions of atoms. Informally, a SWRL rule may be read as meaning that if all the atoms in the antecedent are true, then the consequent must also be true.

SWRL neither supports negated atoms nor disjunction [team 2012].

In SWRL, the predicate symbols can include OWL classes, properties or data types. Arguments can be OWL individuals or data values, or variables referring to them. All variables in SWRL are treated as universally quantified, with their scope limited to a given rule. For example, the concept of person can be captured using an OWL class called Person. The property indicating that a person owns a car can be expressed using OWL object properties hasCar. To classify all car-owner individuals of type Person to also be members of the class Driver, a rule in SWRL can be expressed as follows:

Person(?p) $\wedge$ hasCar(?p, true) $\rightarrow$ Driver(?p)

Person, hasCar and Driver are OWL named classes, ?p is a variable representing an OWL individual. A named individual (name of a person: Marry) in an ontology can also be referred to directly in the rule. For example:

Person(Mary) $\wedge$ hasCar(Mary, true) $\rightarrow$ Driver(Mary)

One of the most powerful features of SWRL is that it allows user-defined built-ins. A built-in is a predicate that takes one or more arguments and evaluates to true if the arguments satisfy the predicate. An equal built-in can be defined to accept two arguments and return true if the arguments are the same. A number of core built-ins for common mathematical and string operations are contained in the SWRL Built-in Submission. New built-ins can be defined and used in the rules. Users can define built-in libraries to perform a wide range of tasks (Ref: from Web). The following is an example SWRL rule using a core SWRL built-in indicating that a person with an age greater than 17 is an adult:

Person(?p) $\wedge$ hasAge(?p, ?age) $\wedge$ swrlb:greaterThan(?age, 17) $\rightarrow$ Adult(?p)

Where “swrlb” is a namespace qualifier with which the core SWRL built-ins are preceded. This rule, when executed, classifies individuals of class Person with an hasAge property value of greater than 17 as members of the class Adult.

3.1.5.3/ PROTÉGÉ - SOFTWARE SUPPORT FOR OWL

Among several knowledge base editing tools, Protégé is a free, open-source tool developed at Stanford Medical Informatics and it has a community of thousands of users (Ref: from Web). Protégé implements a rich set of knowledge-modeling structures as well as actions. It supports the creation, visualization and manipulation of ontologies in various representation formats [Knublauch et al. 2004].

The Protégé-OWL editor enables users to:

- load and save OWL and RDF ontologies

- edit and visualize classes, properties, and SWRL rules

- define logical class characteristics as OWL expressions

- execute reasoners such as description logic classifiers

- and, edit OWL individuals for Semantic Web markup.

Protégé enables one to describe the concepts and relationships in a particular domain as well as build and populate ontologies. In a Protégé tool, ontologies are represented in the range from taxonomies and classifications, database schemas, to fully axiomatized

theories. One of the remarkable points that makes Protégé become a powerful tool is that users can customize interface to be more convenient and friendly. The user interface supports for entering data and creating knowledge models such as a creation of individuals. By using Protégé, users can create a form to edit components (widgets) for each property of classes in ontology, for instance: when we select a value for a certain property, a default text field widget is provided. In addition, plug-ins and Java-based Application Programming Interface (API) are also embedded tin Protégé to build knowledge-based tools and applications [Protege].

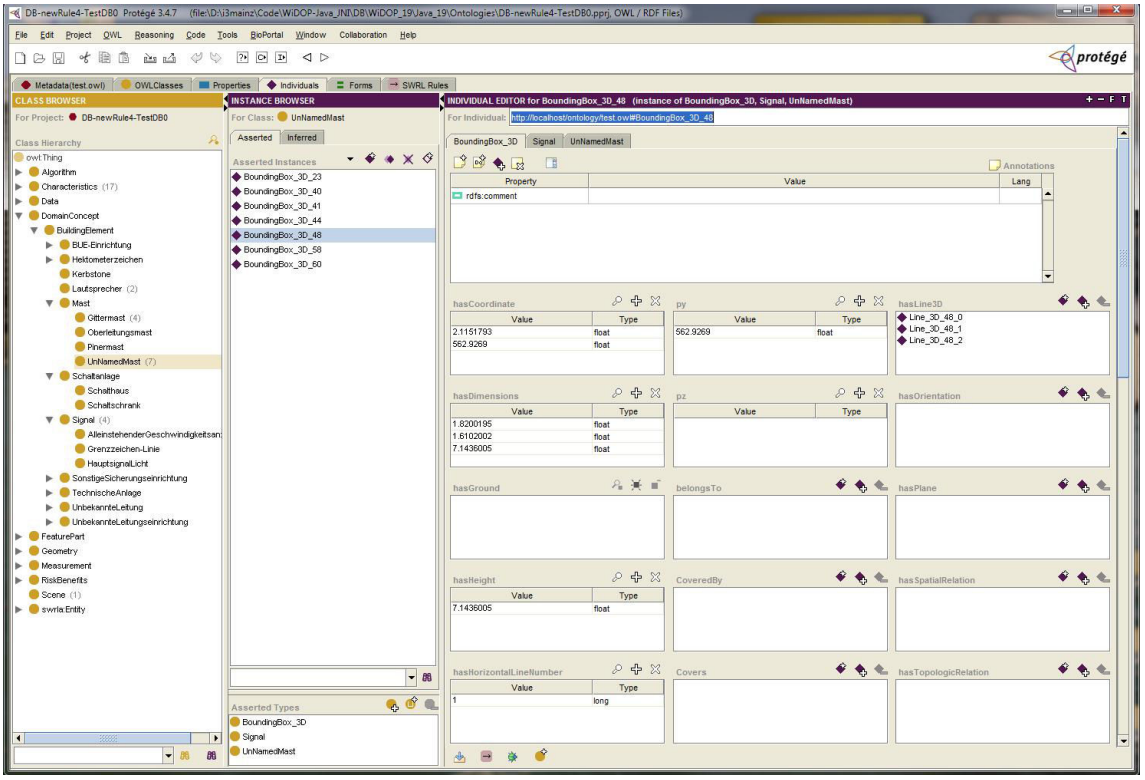


Figure 3.9: OWL represented in the Protégé tool

In recent years, many business and scientific communities have used ontologies as a way to share, reuse and process domain knowledge in the related fields. Gradually, ontologies appear more often in the applications of some fields such as scientific knowledge portals, information management and integration systems, electronic commerce and semantic web services. Several particular features of Protégé distinguish this tool from the other knowledge base editing tools:

- Intuitive and easy-to-use graphical user interface

- Scalability: Protégé's database back-end loads frames only on demand and uses caching to free up memory when needed. There is virtually no deterioration in performance as you go from several hundred frames to several thousand frames.

- Extensible plug-in architecture: Protégé allows to be extended with plug-ins tailored for one's domain and task.

3.2/ NUMERICAL PROCESSING

3.2.1/ DATA ACQUISITION

The modeling of real-world scenarios through capturing 3D digital data is generally carried out by means of a laser scanner. 3D laser scanners nowadays have the ability to quicly and accurately take point measurements on surfaces and landscapes. The representation of objects is digitalized in the form of point clouds in which point's coordinates are the essential elements. Some scanners can acquire RGB information which adds color information to the scanned point clouds. Besides, some scanners can also record intensity data that indicates different states of the point cloud at the scanned area or object surface. Software then analyzes and processes the point clouds to carry out tasks such as registration, normalization, segmentation, etc., to serve the desired targeted application. The use of terrestrial scanner devices for surveying has contributed and advanced the precision and completeness of data acquisition [Bornaz and Rinaudo 2004]. We introduce two scanners, Terrestrial Laser Scanner and LIMEZ [Schewe et al. 1999], which we have used in our work to acquire point clouds.

3.2.1.1/ TERRESTRIAL LASER SCANNER

Figure 3.10: Terrestrial Laser Scanner

A Terrestrial Laser Scanner is able to acquire landscapes point could representations based on laser technology. Laser scanners rely upon the triangulation principle and a high degree of precision (less then 1mm). Terrestrial laser scanning technology is based on Light Detection and Ranging (LiDAR) that is an active imaging system. A laser beam is emitted by the scanner to an object and reflected by the surface of object being scanned. A Terrestrial laser scanner sweeps a laser beam over objects in the scene and records millions of 3D points in a few minutes. The scanner provides highly accurate data which intelligibly represent the real scene in rich 3D point clouds. The result of the laser scanner is a very dense point cloud in which each 3D point is determined by its X, Y and Z coordinates with respect to some given coordinate system and/or additional information such as intensity or color. The notable feature of terrestrial laser scanner is its ability

to provide high resolution 3D maps and images of a scene with centimeter precision. This allows to precisely detect small details (centimeter squares) in a scanned data of such large scenes (kilometers squares). These X, Y, Z measurements (and intensity, color information) can be imported into CAD software such as AutoCAD, 3D point clouds processing applications, etc. to view, measure or be analyzed on a computer.

Figure 3.11: Point cloud of a room

A terrestrial laser scanner is a powerful geodetic imaging tool ideal for aiding users to capture different environments for applications. This equipment is today widely used and gained popularity in various fields such as architectural, archaeological and environmental surveying due to its versatility in use.

3.2.1.2/ LIMEZ III (LICHTRAUMPROFIL MESSZUG)

Growing with technology, some devices for capturing 3D scene have been introduced to serve particular fields of surveying. Besides two types of popular scanners: terrestrial laser scanners and airborne laser scanners. LIMEZ is a typical device that was produced to capture scene in the rail network. LIMEZ is a laser scanner that is usually mounted in a train and captures certain areas of interest in the rail system. LIMEZ produces outputs such as point clouds and images which represent area within facilities of the rail network and obstacles. This device is used to enforce safety such as in obstacles detection.

LIMEZ III is launched as a new clearance profile recording train (serves for the Deutsche Bahn AG, Germany) in September 2006 [Hoefler et al. 2006]. The features are like high speed video techniques, photogrammetry, light sheet technology, forward view laser scanning and fast side view laser scanning, all of which are compacted in the LIMEZ III and overcome the drawbacks of its predecessor.

LIMEZ III consists of two majors subsystems:

- Side view laser scanners for profile measurement: Two fast laser scanners measure

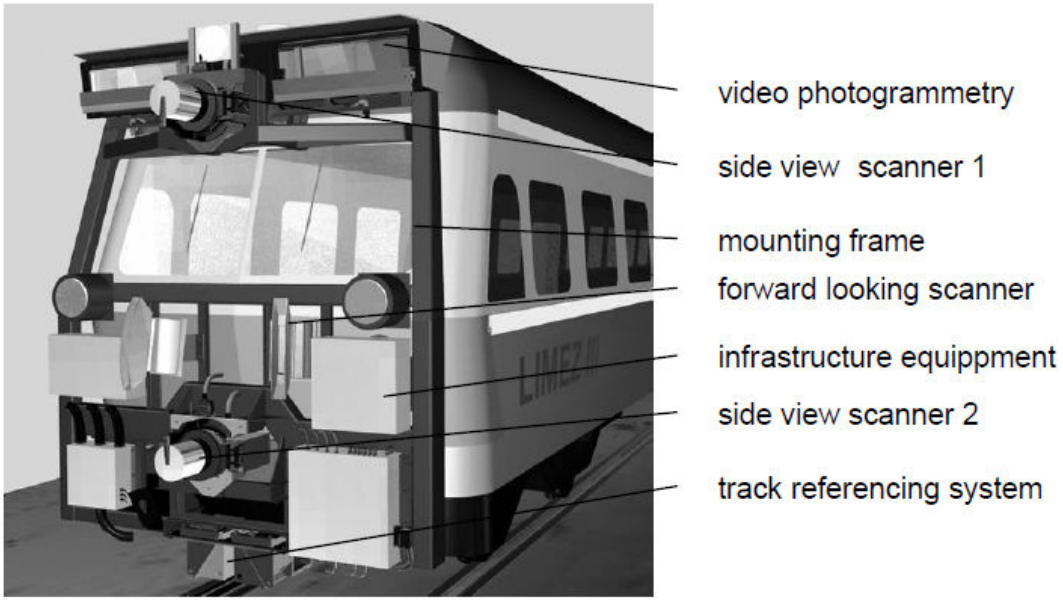


Figure 3.12: LIMEZ III measurement system [Hoefler et al. 2006]

almost 1200 clearance profiles per second with 3600 pixels each. Each laser scanner comprises a 45 degrees tilted and rotating double sided mirror. Two laser beams reflected on its surfaces propagate into opposite directions. The distance between two profiles increases up to 25 mm at a train speed 100km/h.

- Video photogrammetry for supplement measurements and documentation: comprises four 2-megapixel monochrome video cameras which are protected against high temperature conditions. Both cameras are attached on both right and left side of a mounting frame to capture pictures from the right and the left part of the measurement range.

Besides, there are other subsystems added to support the entire system such as: track measuring system for referencing the other system to the track, wide view scanners for forward scanning and identification of possible infringements and inertial navigation and positioning system for measuring all vehicle movements for train localization.

All point clouds acquired from the scanners are registered automatically to the track by the track referencing system. The closest scan profile therefore can be compared in real time to reference clearance profiles. LIMEZ III acquires large amounts of data and is transferred to an available online data preprocessing. The processed data are saved to hard disks and also stored additionally on the vehicle on a network attached storage (capacity of 5 TB) to avoid data loss during transport.

3.2.2/ NOISE REDUCTION

The presence of noise in point cloud datasets may be due to various reasons. For instance, noise can be caused by scanning conditions adequate. Best scanning results are obtained when the surface of the scanned object is perpendicular, or at least close, to the laser beam. In addition, the surface materials may affect the refection of laser beam resulting in 3D points of dubious quality. Noise in data leads to erroneous values and causes representation failures of objects in point clouds [Soudarissanane et al. 2008].

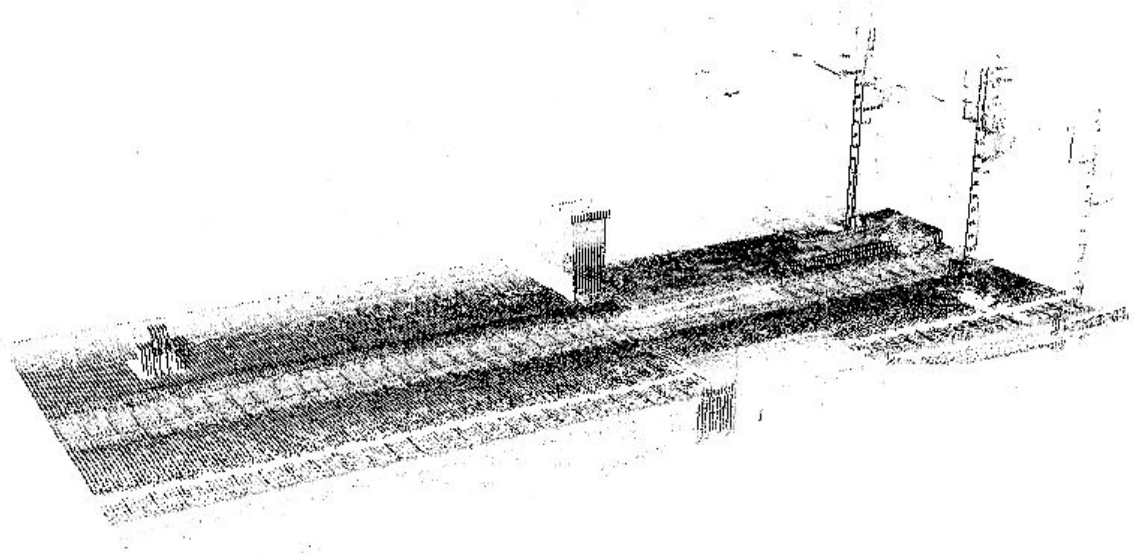


Figure 3.13: Point cloud data of a railroad segment acquired by the LIMEZ III

One way to reduce the effect of noise is to is based upon the computation of the distribution of point-to-neighbors distances in the dataset [Rusu and Cousins 2011]. By assuming that the distribution of such distances in the input data is Gaussian, a mean distance from each point to its surrounding points and a standard deviation are calculated. A point is considered as an outlier (and eliminated from the dataset) if its mean distance is out of an interval defined by the global distances mean and standard deviation.

3.2.3/ Fitting 3D points to primitive shapes

When object detection is considered, bottom-up approaches often begin from the detection of individual base elements of the object. The greater details are then identified by combining and/or deducing from the detected elements. Most objects are modeled by primitive shapes such as planes, circles, linear structures, etc. In the following, two basic methods often used for extracting linear and planar structure are introduced.

3.2.3.1/ Fitting 3D points to a plane (Orthogonal Distance Regression Plane)

Given a set of 3D points, fitting points to a plane is carried out through Orthogonal Distance Regression. It consists in determining the parameter of a hypothesis plane by minimizing the distance from the points to the plane.

Assume that we need to calculate the distance from $P_1(x_1, y_1, z_1)$ to a plane which has formula: $ax + by + cz + d = 0$ and normal vector $\vec{v}(a, b, c)$. $H(x, y, z)$ is a point on the plane, and we call vector $\vec{HP}$ is $\vec{w}$. Projecting PH onto $\vec{v}$ gives the distance h between Q and H of the point P to the plane:

$$|v.w| = |v|.|w|.cos(\alpha) = |v|.QH \tag{3.1}$$

$$h = \frac{|w.v|}{|v|} = \frac{|a(x_1 - x) + b(y_1 - y) + c(z_1 - z)|}{\sqrt{(a^2 + b^2 + c^2)}} \tag{3.2}$$

$$= \frac{|ax_1 + by_1 + cz_1 - (ax + by + cz)|}{\sqrt{(a^2 + b^2 + c^2)}} \tag{3.3}$$

Since we have $ax + by + cz + d = 0$, then $d = -(ax + by + cz)$, it leads to:

$$QH = \frac{|ax_1 + by_1 + cz_1 + d|}{\sqrt{(a^2 + b^2 + c^2)}} \tag{3.4}$$

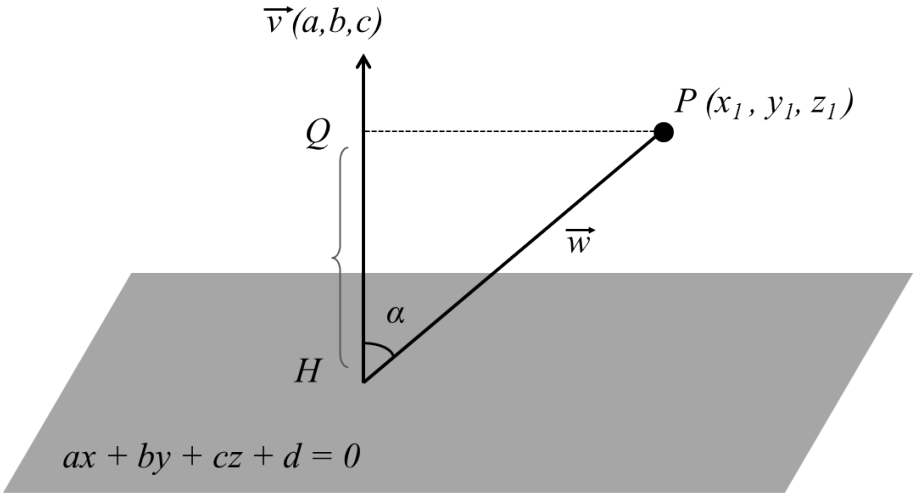


Figure 3.14: Distance from a 3D point to a plane

Calculating the squared distances from every point $P_i(x_i, y_i, z_i)$ to the plane, we have an equation:

$$f(a, b, c, d) = \sum \frac{|ax_i + by_i + cz_i + d|^2}{(a^2 + b^2 + c^2)} \tag{3.5}$$

A point is considered as on the plane, if its distance to the plane approaches zero. Therefore, the purpose is to minimize the sum of squared distances to the plane in order to find a, b, c and d.

In order to derive d, we set the partial derivative f with respect to d equal to zero:

$$\frac{pf}{pd} = 2 \sum \frac{(ax_i + by_i + cz_i + d)}{(a^2 + b^2 + c^2)} = 0 \tag{3.6}$$

$$\sum (ax_i + by_i + cz_i + d) = 0 \tag{3.7}$$

$$a \sum x_i + b \sum y_i + c \sum z_i + Nd = 0 \tag{3.8}$$

where N is the number of points,

$$Nd = -(a\sum x_i + b\sum y_i + c\sum z_i) \tag{3.9}$$

$$d = -\frac{(a\sum x_i + b\sum y_i + c\sum z_i)}{N} \tag{3.10}$$

$$d = -(a\frac{\sum x_i}{N} + b\frac{\sum y_i}{N} + c\frac{\sum z_i}{N}) \tag{3.11}$$

If $P_o(x_0, y_0, z_0)$ is the centroid of the data, then we have:

$$d = -(ax_0 + by_0 + cz_0) \tag{3.12}$$

We substitute d into the plane equation, the equation is then transformed as:

$$a(x - x_0) + b(y - y_0) + c(z - z_0) = 0 \tag{3.13}$$

Now, $f(a, b, c, d)$ can be rewritten as following:

$$f(a, b, c) = \sum \frac{|a(x_i - x_0) + b(y_i - y_0) + c(z_i - z_0)|^2}{(a^2 + b^2 + c^2)} \tag{3.14}$$

Before representing the equation as a matrix, we define v as a vector consisting of three components a, b and c:

$$v^T = \begin{bmatrix} a & b & c \end{bmatrix}$$

and M such that:

$$M = \begin{bmatrix} x_1 - x_0 & y_1 - y_0 & z_1 - z_0 \\ x_2 - x_0 & y_2 - y_0 & z_2 - z_0 \\ . & . & . \\ . & . & . \\ . & . & . \\ x_n - x_0 & y_n - y_0 & z_n - z_0 \end{bmatrix}$$

If we multiply the matrices out, we see that f(a,b,c) becomes

$$f(v) = \frac{(v^T M^T)(Mv)}{(v^T v)} \tag{3.15}$$

$$= \frac{v^T (M^T M)v}{(v^T v)} \tag{3.16}$$

Let's define $A = M^T M$, then $f(v)$ is on the form of the Rayleigh Quotient. It is minimized by the eigenvector of A that corresponds to its smallest eigenvalue.

Minimum $f(v)$ = smallest eigenvalue of A when v = eigenvectors of A

The Singular Value Decomposition (SVD) of M is:

$$M = USV^T$$

where S is a diagonal matrix containing the singular values of M, the columns of V are its singular vectors, and U is an orthogonal matrix.

Replacing $M = USV^T$ in $A = M^TM$, we have:

$$A = (USV^T)^T(USV^T) \tag{3.17}$$

$$= (VS^TU^T)(USV^T) \tag{3.18}$$

$$= VS^2V^T \tag{3.19}$$

The eigenvalues of A are the squares of the singular values of M, and the eigenvectors of A are the singular vectors of M. The Orthogonal Distance Regression Plane contains the centroid of the data, and its normal vector is the singular vector of M corresponding to its smallest singular value [George 2005].

3.2.3.2/ Fitting 3D points to a line (Orthogonal Distance Regression Line)

Fitting the given 3D points to a line is also based on the Orthogonal Distance Regression. In this case, however, it consists in computing the parameter of a hypothesis line in 3D by minimizing the distances from the points to the line. A line L can be parameterized such as:

$$x = x_0 + a.t$$
$$y = y_0 + b.t$$
$$z = z_0 + c.t$$

where (x_0, y_0, z_0) is a point on the line and (a, b, c) is a unit vector.

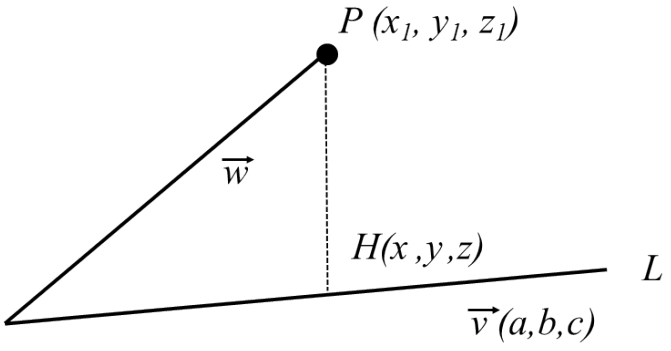


Figure 3.15: Distance from a 3D point to a line

We calculate the distance from $P(x_1, y_1, z_1)$ to the line L. Projecting P onto the line, we have the distance h between P and H, where $H(x, y, z)$ is a point on the line L. Squared distance of P to the line is calculated as:

$$h^2 = (x_1 - x)^2 + (y_1 - y)^2 + (z_1 - z)^2 \tag{3.20}$$

$$= (x_1 - x_0 - a.t)^2 + (y_1 - y_0 - b.t)^2 + (z_1 - z_0 - c.t)^2 \tag{3.21}$$

Substitute

$$t = \frac{(x - x_0)}{a} = \frac{(y - y_0)}{b} = \frac{(z - z0)}{c}$$

we have:

$$h^2 = (x_1 - x_0 - \frac{a(y - y_0)}{b})^2 + (y_1 - y_0 - \frac{b(z - z_0)}{c})^2 + (z_1 - z_0 - \frac{c(x - x_0)}{a})^2 \tag{3.22}$$

$$= \frac{[b(x_1 - x_0) - a(y - y_0)]^2}{b^2} + \frac{[c(y_1 - y0) - b(z - z_0)]^2}{c^2} + \frac{[a(z_1 - z_0) - c(x - x_0)]^2}{a^2} \tag{3.23}$$

To minimize the sum of squared distances from every point Pi to the line, we minimize a sum of squared distances as follows:

$$d = \sum [b(x_i - x_0) - a(y–y_0)]^2 + [c(y_i - y_0) - b(z - z_0)]^2 + [a(z_i - z_0) - c(x - x_0)]^2 \tag{3.24}$$

We take derivatives with respect to x_0, y_0, and z_0 and set the results equal to zero, we get equations that can be manipulated to yield:

$$\frac{(x_0 - \bar{x})}{a} = \frac{(y_0 - \bar{y})}{b} = \frac{(z_0 - \bar{z})}{c} \tag{3.25}$$

where $(\bar{x}, \bar{y}, \bar{z})$ is the centroid of the data.

To find vector (a, b, c), we consider C as the centroid, L as the Orthogonal Distance Regression Line and Q as the plane through C such L is perpendicular to P.

For a set of points Pi, sum of distances from Pi to line L is calculated based on the Pythagorean theorem:

$$\sum [distance(P_i, L)^2] = \sum [distance(P_i, C)^2] - \sum [distance(P_i, Q)^2]$$

Note that, the unit vector v is normal vector of plane Q. Seeking (a, b, c) that minimize $\sum [distance(P_i, L)^2]$ is equivalent to maximizing $\sum [distance(P_i, Q)^2]$ because $\sum [distance(P_i, C)^2]$ is a constant. We recall that the minimum distance from a point to a plane is found by choosing the singular vector of M that corresponds to its smallest singular value. For the Orthogonal Distance Regression Line, the maximum distance from

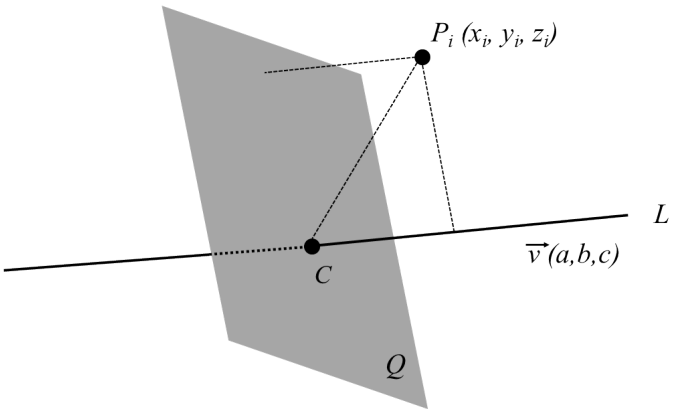


Figure 3.16: A plane Q is assumed to compute the minimum distance from P to line L

a point to the plane is sought, we thus want the eigenvector of A that corresponds to its largest eigenvalue, or the singular vector of M that corresponds to its largest singular value.

The 3D Orthogonal Distance Regression Line contains the centroid, and its direction vector is the largest eigenvector (or singular vector of M) corresponding to its largest singular value [George 2005].

3.2.4/ 3D TO 2D PROJECTION

A projection is a way of transforming an object from one dimensionality to another. When the 3D world is transformed to 2D one, the world is projected on a finite plane. We distinguish in the following between orthogonal projections and perspective projections [McMillan 2005], [Hearn et al. 2010].

Orthogonal projection

An orthogonal projection (also called a parallel projection) is simple projection that preserves the object's measurements in two dimensions. However, the projection of the object does not appear natural lacks perspective.

The projection matrix for orthogonal projection expresses as:

$$P_{orthogonal} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.26}$$

The coordinates (x, y, z) are transformed to the new ones (x', y', z') as:

$$\begin{bmatrix} x' & y' & z' & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x & y & z & 1 \end{bmatrix} = \begin{bmatrix} x & y & 0 & 1 \end{bmatrix} \tag{3.27}$$

In general, an orthogonal projection has some features such as:

- Scale is preserved;

- Good for exacting measurements;

- Angles are not preserved;

- Has no vanish-point;

- Parallel lines remain parallel.

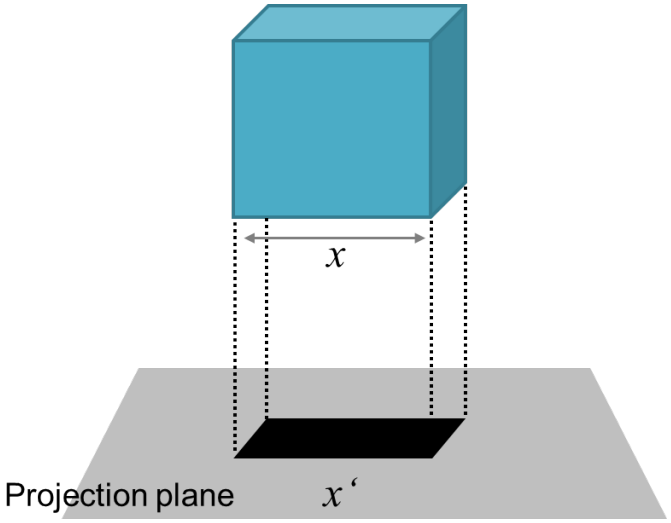


Figure 3.17: An example of an orthogonal projection of a cube on a horizontal plane

Perspective projection

The perspective projection differs from the orthogonal projection in the way it simulates perspective viewing. The perspective project allows us to denote the foreshortening of objects in 3D world. When applying a perspective projection for the lines in 3D, they always intersect at a point. Such transformation therefore does not preserve the measurements of object.

The projection matrix for perspective projection matrix is:

$$P_{perspective} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/D & 0 \end{bmatrix} \tag{3.28}$$

Where D is distance from the projection plane to the projection reference point. The transformed coordinates (x', y', z') of a point 3D (x, y, z) is computed:

$$\begin{bmatrix} x' & y' & z' & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/D & 0 \end{bmatrix} \begin{bmatrix} x & y & z & 1 \end{bmatrix} = \begin{bmatrix} x & y & z & z/D \end{bmatrix} \tag{3.29}$$

This leads to:

$$x' = \frac{x.D}{z} \tag{3.30}$$

$$y' = \frac{y.D}{z} \tag{3.31}$$

and

$$z' = D \tag{3.32}$$

(z component is omitted in 2D)

In general, a perspective projection has several features such as:

- Making close objects seem bigger;
- Have vanishing points;
- Not preserving distances and angles;
- Parallelism is not preserved.

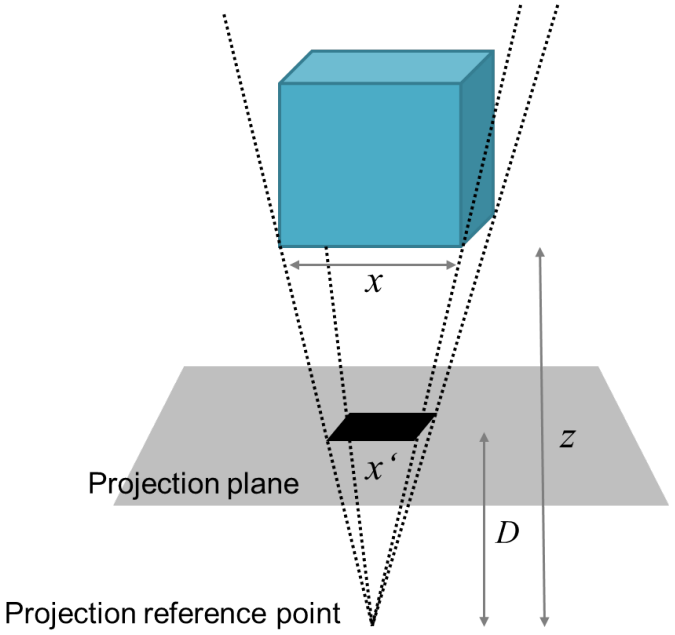


Figure 3.18: An example of a perspective projection of a cube on a horizontal plane

3.2.5/ RANSAC

The RANdom SAmple Consensus (RANSAC) algorithm was proposed by Fischler and Bolles in 1981 [Fischler and Bolles 1981], is an iterative method for estimating parameters of a mathematical model from a set of data. RANSAC uses the smallest set possible and proceeds to enlarge this set with consistent data points.

A mathematical model estimated by iteratively selecting a random subset of the original data which are hypothetical inliers. A model is then fitted to the hypothetical inliers and the parameters of this model are determined. RANSAC algorithm is described as the following:

1. A minimum number of points is selected to determine the model parameters.
2. Solve for the parameters of the model.
3. Find points from the dataset which satisfy a pre-defined tolerance. These points are inliers. (for example in case of line fitting, a point fits the pre-defined tolerance if the distance t from it to the line is smaller than a threshold).
4. If $N_{inliers}/N_{points}$ exceeds a pre-defined threshold, re-estimate the model parameters using all the identified inliers and terminate.
5. Otherwise, repeat steps 1 through 4.

Maximum of iteration number is chosen high enough to ensure that at least one of the sets of random samples does not include an outlier [Derpanis 2010].

Pros:

- Simple and general

- Applicable to many different problems

Cons:

- Require to tune many parameters

- Require many iterations to obtain a best result

- Time consuming.

3.2.6/ Point Cloud Library

The Point Cloud Library (PCL) (www.pointclouds.org) is built as a comprehensive free library for n-D Point Clouds and 3D geometry processing. PCL is a standalone, large scale, open project for 2D/3D image and point cloud processing. This library is written in C++ and optimized for solving numerical processing problems. State-of-the art algorithms for n-dimensional point clouds and 3D geometry processing are implemented in the PCL. The library allows using algorithms for filtering, feature estimation, surface reconstruction, registration, model fitting and segmentation.

The algorithms in PCL are grouped to the smaller libraries [Rusu and Cousins 2011]:

- *libpcl io:* provides input/output operations such as writing to/reading from Point Cloud data files;

- *libpcl registration:* implements point cloud registration methods such as ICP, etc;

- *libpcl filters:* contains algorithms such as down sampling, outlier removal, indices extraction and projections, etc;

- *libpcl features:* provides algorithms to extract features such as surface normal vectors and curvatures, boundary point estimation, moment invariants, PFH and FPFH descriptors, NARF descriptors, RIFT, SIFT on intensity data, etc;

- *libpcl segmentation:* consists of cluster extraction, model fitting methods for a variety of models such as planes, cylinders, spheres, lines, etc.

- *libpcl keypoints:* implements different keypoint extraction methods;

- *libpcl range image:* contains tools that support range images created from point cloud datasets.

- *libpcl surface:* includes algorithms based on surface reconstruction techniques, meshing, convex hulls, Moving Least Squares, etc;

PCL has been developed thanks to the contribution of a large number of researchers and engineers around the world. This library has been ported to Windows, MacOS, Linux, and Android. As it has been developed by a community, the library has expanded over time. PCL is used as an independent library in a framework which serves for different purposes.

4

METHODOLOGY

4.1/ SYSTEM OVERVIEW

When attempting to build an integrated approach with knowledge directing all parts of the process, several aspects have to be considered. At first, the whole process needs to be incorporated into a knowledge management tool. Therefore, it is necessary to have a process guiding all individual steps, leading from an initial situation to the final result. Inside this overall process, one part has to cover the numerical processing and another part has to handle the processing results. This latter part has to evaluate the results, draw conclusions about what has been found, and also what this means for further processing. This includes the need to update the content of the database with the objects that have been found. This database has to be managed in a way that every detected object is transferred from some initial state to a final one within the framework of a rule-based system.

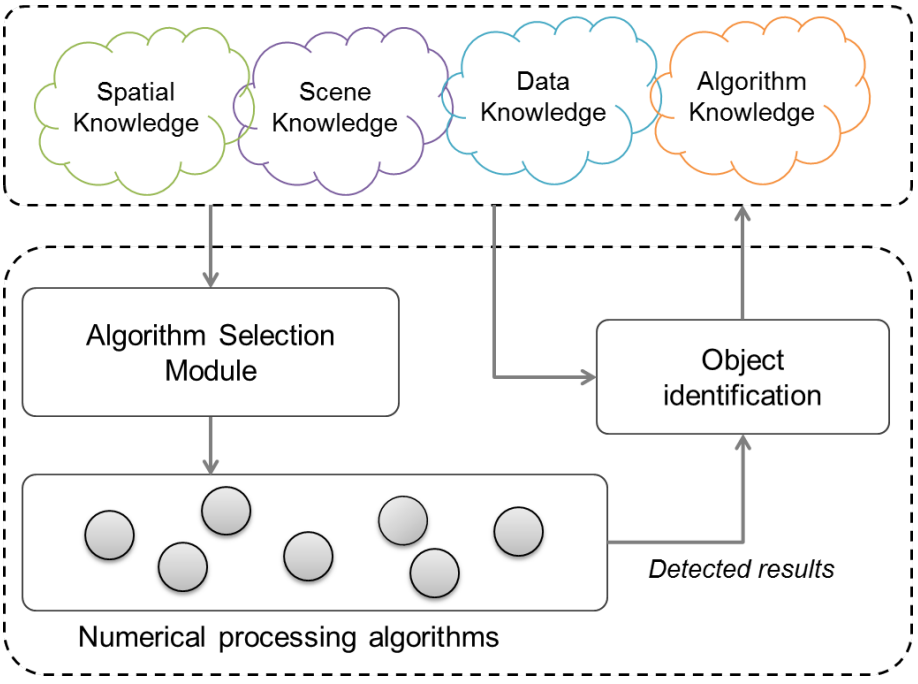


Figure 4.1: System architecture

The main components of our system are illustrated in Fig. 4.1. The strategy is applied

to the analysis of 3D point clouds, but can also be extended to other data sources. It is based on explicitly formulating prior knowledge of the scene, on spatial relations of objects and on processing algorithms. It is a multi-stage concept based on three components: the modeled knowledge Fig. 4.1 (left), the package of algorithms Fig. 4.1 (right) and the classification engine Fig. 4.1 (bottom-right). In the initial stage, the available knowledge is transferred into a knowledge base. Starting from this initial stage, an updating process, which invokes the algorithms and the classification engine, is launched. Here, the algorithm selection module guides the processing via selecting a set of processing algorithms based on the nature of the target objects, and produces new elements which can be identified. These elements are passed on to the classification engine, which, based on the existing knowledge expressed in the ontology, attempts to apply Semantic Web Rule Language (SWRL) rules [Horrocks et al. 2004] and Description Logic (DL) constraints in order to identify the nature or object category of the elements. This classification handles the output obtained from the algorithms. The result of the classification step updates the knowledge base by inserting newly classified elements or updating already existing elements before running the next stage of processing. The process ends either when all objects are detected and classified or when the annotation process stalls after a predetermined number of iterations.

Objects are represented by a point cloud (or possibly data from other sources). Such data depend on many factors such as the type of the sensing system and the measuring/capturing conditions. This representation has to be handled by algorithms which also depend on many additional factors (e.g. noise, other data characteristics, and already existing objects). Strong interrelationships among these factors have a direct influence on the efficiency of the detection and classification processes. The more flexibly these factors and interactions are controlled, the better results are to be expected. For these reasons, knowledge from different domains is required and the quality of these various knowledge sets has significant impact on the results. Our solution relies on four main knowledge categories to construct the core of the knowledge base: the scene knowledge, the spatial knowledge, the data knowledge and the algorithm knowledge. Each field of knowledge is represented by circles in Figure 1, and relationships between these concepts are represented by edges. The scene knowledge contains information related to the content of the scene to be processed, important characteristics of objects (e.g. geometric features, appearance and texture), and the geometry that composes its structure. Such knowledge is not only important for identification and classification processes but also supports the selection and guidance of the algorithms. The spatial knowledge models the relationships between objects in the scene. It is an important factor for the classification process because it supports an object's state disambiguation based on its relationship with the common environment. The data knowledge expresses important characteristics of the data itself. Finally, algorithm knowledge characterizes the behavior of algorithms and determines which purpose they fulfill, which input is expected, which output is generated, and which geometries they are designed for. Based on this knowledge, a dynamic algorithm selection is possible allowing for a dynamic adaptation to processing situations given from other domains Fig. 4.1.

4.2/ KNOWLEDGE ENGINEERING

Our concept requires efficient methods for knowledge representation, management and interaction with algorithms. An ontology is a formal representation of knowledge as a set of concepts within a domain, and the relationships between those concepts. It is used to reason about the entities within that domain, and may be used to describe the domain. In theory, conventionally, ontology presents a ”formal, explicit specification of a shared conceptualization”. An ontology provides a shared vocabulary, which can be used to model a domain. Well-made ontology owns a number of positive aspects like the ability to define a precise vocabulary of terms, the ability to inherit and extends exiting ones, the ability to declare relationships between defined concepts and finally the ability to infer new relationships by reasoning on existing ones [Gruber et al. 1993].

4.2.1/ KNOWLEDGE MANAGEMENT TECHNIQUES

OWL

Efficient knowledge representation tools are available from the Semantic Web framework, which expresses knowledge through the Web Ontology Language (OWL) [Bechhofer et al. 2004]. The encapsulation of semantics within OWL through Description Logics (DLs) axioms has made it an ideal technology for representing knowledge from almost any discipline. We use the OWL to represent expert knowledge about the scene of interest and for algorithmic processing. With OWL ontology, we are able to describe complex semantics of a scene.

For instance, the statement “A railway track is a linear feature with two linear structures running parallel to each other within a certain distance” can be expressed through logical statements. Likewise, we define the semantics of algorithmic processing within OWL. For example, the *CheckParallel* algorithm is designed for detecting a *Signal*, which contains parallel linear structures.

$$CheckParallel \exists isDesignedFor.Signal \sqcap Signal.hasParallel.\{true\} \tag{4.1}$$

SWRL

Despite the richness of OWL's set of relational properties, the full range of expressive possibilities for object relationships is not covered yet. Besides technologies known as Semantic Web, most precisely the OWL, an additional technology is SWRL. It is a program which infers logic from the knowledge base to derive a conclusion based on observations and hypotheses. SWRL allows to declare relationship in term of conditions or rules. Rules perform particular operations on knowledge bases like the consistency checking, the satisfiability checking and finally the expansion of relationships between objects inferred from explicitly stated relationships. The SWRL is open and flexible and allows to integrate Built-Ins, which in our case give access to the world of geometrical processing.

Example 1: A rule asserts that the combination of the hasParent and hasBrother properties implies the hasUncle one. This rule could be written as:

$$hasParent(?x_1, ?x_2) \wedge hasBrother(?x_2, ?x_3) \rightarrow hasUncle(?x_1, ?x_3) \tag{4.2}$$

Where x_1, x_2 and x_3 present individuals from the class Person defined in the ontology and hasParent, hasBrother and hasUncle presents data properties in the same cited structure. As seen in the above example, rules are divided into two parts, antecedent and consequent separated by the symbol "→". If all the statements in the antecedent clause are determined to be true, then all the statements in the consequent clause are applied. In this way, new properties like hasUncle in our example can be assigned to individuals in the ontology based upon on the current state of knowledge base. Add to this standard, SWRL language specify also a library for Built-Ins functions which can be applied to individuals. It includes numerical comparison, simple arithmetic and string manipulation.

Example 2: The following rule (eq. 4.3) asserts that a detected element (of class Geometry) which has a distance from *DistanceSignal* of 1000m, has a height equal to or greater than 4m, and which has a linear structure, will be inferred as a *MainSignal*.

$$\begin{aligned} &Geometry(?x) \wedge hasLine(?x, ?l) \wedge line(?l) \wedge DistanceSignal(?y) \\ &\wedge DistanceFrom(?x, ?y, ?dis) \wedge swrlb : GreaterThan(?dis, 1000) \wedge \\ &hasHeight(?x, ?h) \wedge swrlb : GreaterThan(?h, 4) \rightarrow MainSignal(?x) \end{aligned} \tag{4.3}$$

Variables are indicated by the standard convention in which they are prefixed by a question mark symbol (e.g. $?x$). An important SWRL feature is its ability to allow user-defined built-ins (user-defined predicates, such as, swrlb:equal and swrlb:lessThan, that can be used in SWRL rules) which help in the interoperation of SWRL with other formalisms and provide an extensible infrastructure for knowledge-based applications.

4.2.2/ KNOWLEDGE MODELING

The techniques mentioned above serve as tools to formalize the identified and acquired knowledge. We model an ontology which consists of four major knowledge domains: SceneKnowledge, SpatialKnowledge, DataKnowledge and AlgorithmKnowledge Fig. 4.2. SceneKnowledge contains the object information within the scene, such as definitions and their properties. DataKnowledge expresses important data characteristics (for example, density, noise, and the resolution of data) and is acquired from sources such as available documentation, CAD, Geographic information system (GIS) and expert assistance. SpatialKnowledge contains information about how objects are scattered in the scene, and their spatial relationships represent the geometric relations either among objects in the scene or components in an object. AlgorithmKnowledge includes the algorithm definitions and their properties. SceneKnowledge and AlgorithmKnowledge are linked together through similarities between the object properties (defined in SceneKnowledge) and algorithm properties (defined in AlgorithmKnowledge). In order to gain "intelligence" in processing, the algorithms adapt to specific situations through parameter adjustment. Data characteristics *(low density, high density, distinct, invisible, noise, etc.)* as well as object geometry *(thick, thin, flat, broken, etc.)* are defined in Characteristic. These characteristics usually influence algorithm performance, and the system is able to set algorithm parameters in order to adapt to different object and data characteristics. The system requires an expert to adjust and decide appropriate values (for algorithm's parameters) which are stored in the sub-class RiskBenefit.

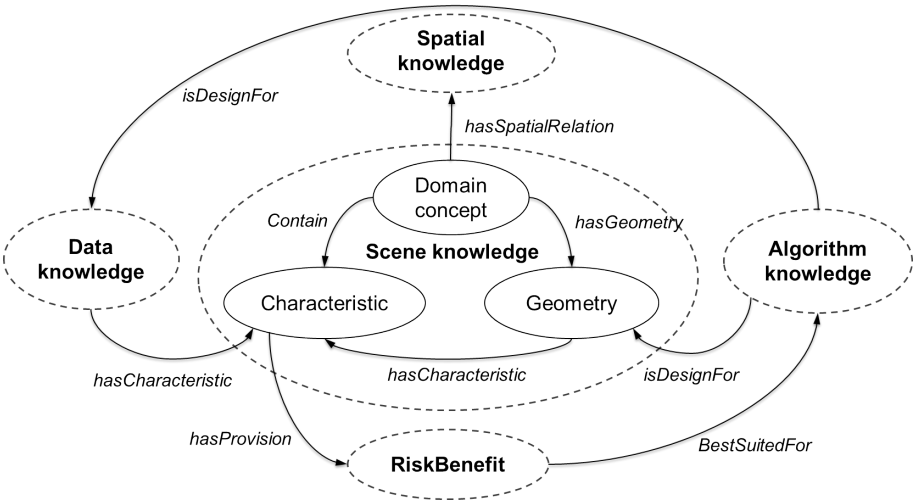


Figure 4.2: General ontology schema overview

4.2.2.1/ SCENE KNOWLEDGE

The scene knowledge is described in the schema of ontology and includes semantics of the objects such as properties, restrictions, relationships between objects and geometries. These elements are managed in the scene knowledge and stored to one in three layer classes: Domain Concept, Geometry and Characteristic. Particularly, Domain Concept contains classes of object of interest in the scene. They are defined by their names and particular features. Geometry shapes of object, such as: linear structure, planar surface, etc., are sorted in the class Geometry. Characteristic class consists of instances which describe characteristics of geometry or data, for example: thin, thick, low density, high density, etc. The more information about an object is created and used, the more accurate the detection and classification process is.

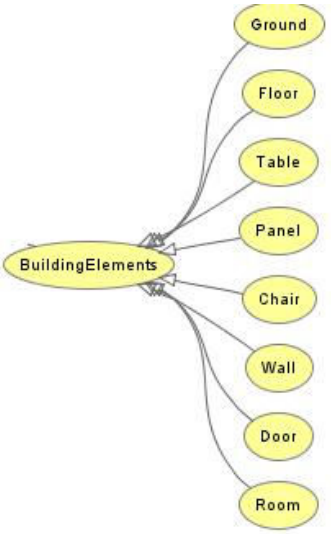


Figure 4.3: Grouping of scene objects in case of a building

Modeling knowledge for a part of a scene can be described as following example. Fig.

4.3 shows classes of object in a building. A collection of classes is represented in Domain Concept. They are additionally structured in a hierarchical order as might be seen convenient for a scene. This could lead to relations like: a room is a super class of wall and floor, with door as further sub class of wall. But also other ordering can be imagined, as a structuring with respect to processing aspects. Such a structure could separate between different complexity of classes. Simply structured objects like walls, ground floors or ceiling then would be distinguished from other objects in accordance to their impact onto the processing strategy. Simple objects require simple detection strategies, whereas complex ones will be composed out of several geometrical objects needing adapted and more complex processing strategies. They first have to be decomposed into their geometric objects, which then have to be verified and regrouped based on known topology relations between them. Likewise, a table as a complex element is composed of a plane representing the table top and at least one linear structure, representing a leg. Once theses geometries are detected, the topology decides upon the correctness of this assumption, as the plane (table top) must be connected and perpendicular to the linear structure (leg). This is just a first draft since modeling depends on the target scene to be detected.

Each one of the above mentioned object classes have relations to the Geometry class. This class handles features which may have an impact or are useful for decisions based on geometrical aspects. It helps to enrich scene objects with additional information or provide data for the processing strategies. Geometry class contains information about the different geometric elements composing a semantic object, like plane, line, sphere and others. A chair for example has linear elements (legs of the chair), a leaning plane and a seat plane (Fig. 4.4).

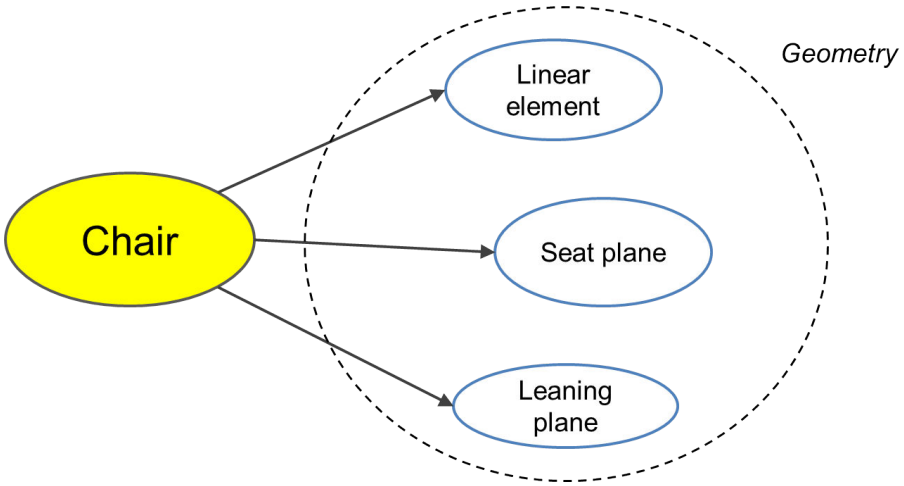


Figure 4.4: The geometry class hierarchy

To specify its semantic characteristics, Characteristic class is created, aiming to characterize a semantic object by a set of characteristics like color, size, visibility and its position in the point cloud after detection. To do so, new object properties like “hasColor”, “hasOrientation”, “hasFeature” are created linking the properties of object class to defined characteristics “Color”, “Orientation”, “Thin” in the Characteristic class respectively (Fig. 4.5).

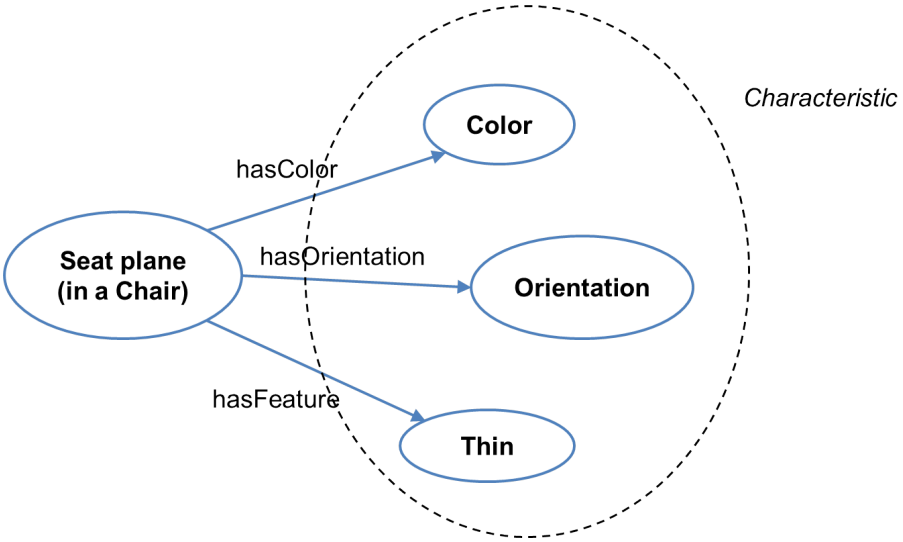


Figure 4.5: Object and data properties characterizing the semantic objects

4.2.2.2/ DATA KNOWLEDGE

There are different kinds of input data that would be used in a 3D processing task: images, range images, 3D point clouds, etc. These data are directly used in extracting objects present in a scene. There are other kinds of data, for example: documents in related domain, CAD, GIS, etc., are also utilized to provide additional information. A collection of necessary resources allows processing to be able to quickly detect/identify targeted objects and achieve reliable results. To do that, we exploit different features of data such as how much data characteristics (in term of data quality, attributes and error in data) influence the numerical processing. For example, an object detection algorithm can succeed or fail at a certain point during the process due to the quality of data. To understand data characteristic, a process is required to extract information from data and draw rules from such information. This process is a knowledge modelling from data, all influences of data to processes are framed as rules which are stored in “Data knowledge” – a class of knowledge base. Depending on the type of data, we have particular method to extract data characteristics. Point cloud data are primarily used in our approach, in the data knowledge modelling we thus focus on this kind of data in some aspects such as quality, measurement error and feature of measurement devices.

First, data quality has a tremendous effect during the execution of an algorithm (for example when detecting objects). In particular, thresholds in the algorithm should be properly set to adapt to the quality of data. We create different levels of data density, for instance low, medium and high. These notions are manually defined based on the number of 3D points per volumetric unit. Threshold in an algorithm used for a low density data is different from one used for a high density data. These rules or relations are defined in “Data knowledge”.

Second, measurement errors are often responsible for algorithms failure. Point cloud data with errors supply a sufficient number of points. However, their coordinates are not faithful to the captured scene. Measurement errors mostly depend on the distance and angle from scanner to surface object. Basically, point clouds representing an object that is far from the scanner have high probability of containing errors. Therefore any result

obtained from such data will be uncertain. By a relative evaluation, we assign probability values for these results indicating how much trust is put into each. The relationship between measurement error and probability of accuracy of the results are modelled in data knowledge.

Third, measurement devices have directly influents to the way we model an algorithm. In the current study, we have used two kinds of laser scanners: a terrestrial laser scanner and LIMEZ III. These devices have been used to capture scenes and produce point cloud data. Using multiple terrestrial laser scanners allows us to have full representation of an object's surfaces, while LIMEZ III supplies only one face of an object. Additionally, data density acquired by this equipment has lower density than terrestrial laser scanner's (in our experiments). The significant distinction between these two scanners requires a modification of the algorithms. For example: two different line extraction algorithms have been designed. One for the data acquired from the terrestrial laser scanner and the other for data acquired from LIMEZ III. Besides, since objects captured by this equipment only show one side, we only focus on algorithms whose features are found from the visible sides.

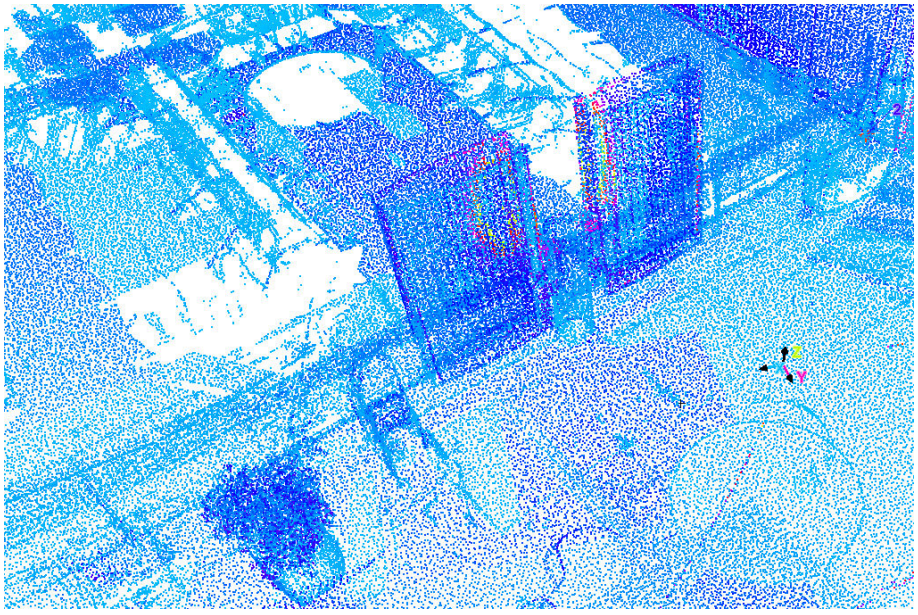


Figure 4.6: A point cloud acquired by Terrestrial Laser Scanners, all surfaces of panels in the scene are captured

All factors originating from data characteristics and having influence on the processing are collected and modelled in the data knowledge. In order to select the best suitable algorithm with respect to different data conditions, relations between data characteristics and algorithms are established to interpret the influences mentioned above. These relations are modelled in forms of knowledge representation which are then stored in Data knowledge.

4.2.2.3/ SPATIAL KNOWLEDGE

Knowledge about 3D spatial relationships is used to enhance the classification process. Information about how objects are scattered in a 3D scene make the detection and clas-

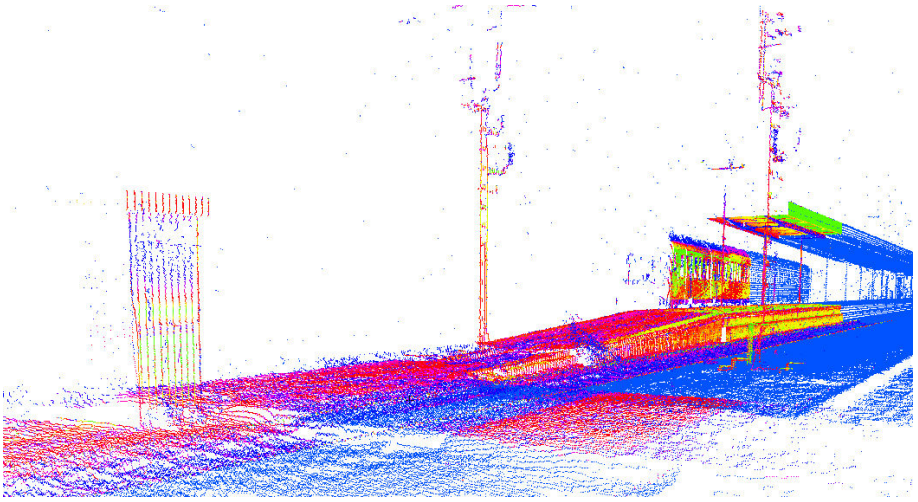


Figure 4.7: Objects scanned by the LIMEZ III only have one face represented in point clouds

sification easier. For instance, given the detection of a wall, there are better chances that a door or window will be detected within it. 3D spatial knowledge includes standards like the 3D topologic knowledge, 3D metric knowledge and 3D processing knowledge. Spatial knowledge contains relationships such as: disjoint, contains, inside, covers, equals, and overlaps. They represent the geometric relationships either between components in an object or between objects in the scene. Each of the mentioned types of spatial knowledge contains a variety of relations modelled in the ontology structure. The top level ontology is designed to include the topological relationships. This is then used to enrich an existing knowledge base to make it possible to define topological relationships between objects in a specific case. Metric knowledge presents important information, because the different elements fulfil very strict metric rules that can also be used in the detection and classification process. In the example of scenes that are specific for railroads, Fig. 4.8 shows an ontological structure, supported by the SWRL rules, which can automatically specify that an object (with certain characteristics) 1000 ± 0.5m away from *Distance signal* can be a *Main signal*.

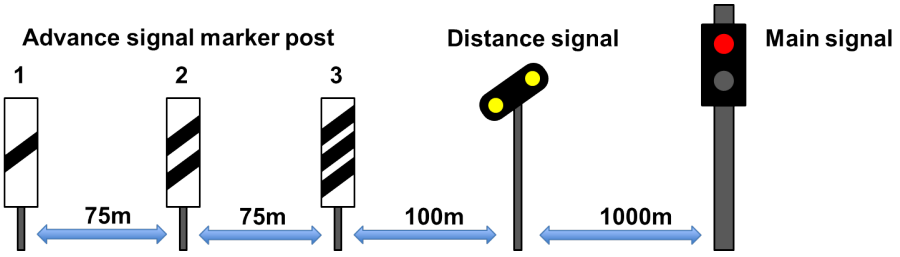


Figure 4.8: Metric rules

Topological knowledge represents adjacency relationships between scene elements. In the case of a building, for example, a topological relation between a wall and the ground floor can be defined as both being connected and the wall must be perpendicular to the ground. The purpose of this class is to spatially connect objects present in the scene and in the geometry layer class. From a semantic point of view, topological properties describe adjacency relations between classes. For example, the property *isParallelTo*

allows to characterize two geometric concepts by the feature of parallelism. Similarly relations like *isPerpendicularTo* and *isConnectedTo* will help to characterize and exploit certain spatial relations and make them accessible during the reasoning steps.

4.2.2.4/ ALGORITHM KNOWLEDGE

The integration of 3D processing algorithms into the semantic framework requires an interaction between scene data and algorithms. Algorithm knowledge is therefore needed to make processing adapt to various conditions of input data (point clouds) as well as different scene. Algorithm knowledge contains all relevant aspects needed to select processing algorithms, generate processing sequences and set parameters for individual algorithms under different situations.

Regarding the numerical processing algorithms, effectiveness depends on the quality of the data (resolution, noise), the characteristics of the object that needs to be detected, or other factors depending on a specific case. Algorithms are modelled under specialized classes of algorithms, sharing certain taxonomical and relational behaviour. The hierarchical representation of the algorithms is addressed by dividing the algorithms according to the context in which they are executed. Likewise, relational semantics are represented by properties. In broader terms, there are two types of relationships: one which applies to the geometry that an object in Domain Concept possesses, and one which relates distinct objects. The first category of relationships is used for detecting geometries. The object property *isDesignedFor* maps algorithms to the respective geometries. For example: *LineDetection isDesignedFor* lines. The second set of algorithm properties *hasInput/hasOutput* are inter-relational properties to connect algorithms based on the compatibility of output from an algorithm to the inputs of others.

To adapt processing to certain situations, depending on the data, the scene and the characteristics of objects, we create a concept that allows for these interactions, as it is able to automatically change the strategy based on a compromise between quality and risks. A part of the knowledge base is dedicated to risk-benefit factors that have an influence on the algorithms and have been deduced from “trial-and-error” simulations on individual algorithms. Since an algorithm may perform best with some given parameters in one setting and fail to deliver the same quality in other settings, it is important to assess the risk-benefit factors of every algorithm with various possible settings. The class RiskBenefit includes all identified risks and benefits. The class contains instances such as *Distinct, Illusive, Noise*, and *DetectionError*. These instances are either the risks or the benefits that have some influence on the algorithms as a whole, or at least the values of the parameters they contain. Note that the classes mentioned above form the general structure of the ontology. They can also be used for other domains, for example: building semantic annotated maps by a mobile robot, mobile mapping of street furniture or forest, semantic place labelling from airborne laser data, etc. In particular, every entity within each class can be replaced by an appropriate one from the application field.

Knowledge modelling and human interaction: The process of modelling knowledge requires the user to collect “information” from related domains. This process is currently carried out manually. “Collecting information” can imply extracting knowledge from various sources or filling the ontology with objects corresponding to specific classes, object properties, algorithms, algorithmic properties, etc. Some of these tasks (such as data

extraction from technical documents) have the potential to be done automatically using specialized processing tools borrowed from the Document Analysis community [Tang et al. 1996]. Depending on the available tools and target application (including its related domains), the knowledge modelling process may take a single person from one to several days of work (data extraction and ontology modelling) including interaction with domain experts and modelling all relationships. Examples of the length of this process and the amount of human interaction are given in next chapter (use cases). However, although such figures may seem significant, one has to keep in mind that knowledge modelling for a given application is done only once and used for processing numerous point clouds with virtually very little or no changes to the ontology. It is also to be noted that other approaches, such as those based on machine learning, would also require a significant amount of preparation to extract training data and carry out annotations generally from large amounts of scans, which may require at least as much time as modelling an ontology. This is especially true when dealing with special environments (such as railroads, industrial plants, etc.) which are often subject to various kinds of regulations, requiring a certain level of expertise.

4.3/ NUMERICAL PROCESSING

The main objective of the set of algorithms provided by the numerical part is to allow detecting geometries belonging to objects. The geometries represent particular features between objects: this is how a human in the first place can recognize objects. In general, the shape of object is represented through geometries and they are composed in a structure that human is able to identify object. Hence, providing the best quality of geometry detection algorithms is crucial for the work we had to carry out on the numerical processing. This work is either based on existing techniques or improved from the state-of-the-art methods to serve our particular targets. The algorithms should provide their efficiency when detecting geometries in different situations (data condition, object characteristic, etc.). This is can be done by changing parameters in each algorithm to adapt with the diversity in the nature of data as well as object characteristics.

The goal of the numerical processing part is to provide a large set of individual algorithms. Each algorithm plays a different role in data processing and some of them can be appropriately linked together in order to complete a certain task. These algorithms are introduced in more detail in the following section.

4.3.1/ ALGORITHM CATEGORIES

We set up the primary groups of algorithms of 3D object detection in point clouds and images that were built to serve different desired targets in our approach. Each particular algorithm corresponds to a particular purpose and data characteristics. Those characteristics can be based on geometric or on radiometric information. With a large variety of different object types of diverse complexity, we need a collection of algorithms. In order to manage them, we classify them into individual groups. This is to structure the “toolbox” in order to make the algorithms available for easy access under the guidance of knowledge. Basically, there are four main groups of algorithms that we classify based on their function:

Group I: "Data processing" specializes in data processing for enhancing the quality of datasets. This group consists of algorithms that allow to sample images and point clouds (or reduce the weight of data) as to lower the unnecessarily high density of points in some areas. We also consider here algorithms designed for reducing noise caused by the limitations in the scanner accuracy or caused by other factors. Regarding 2D imaging data, we have included in this group those algorithms that allow to process binary and color images dealing with matters such as noise reduction, thinning, mathematical morphology operations, edge detection, sharpening edge boundaries and so on.

Group II: "Segmentation" is usually used for separating the regions of data based on certain features. We mainly consider the algorithms which are able to crop point clouds in 3D and specify the segmented region. The primary purpose of algorithms in this group is to segment and then use points representing the objects of interest. The segmented region is determined by a bounding box which contains points surrounding a relative object position (central coordinate of object) and within a defined volume.

Group III: "Geometry detection" contains the primary algorithms that will be used to detect and recognize the primitive shapes or feature of objects. The main interests are geometries such as linear structures, planar structures, points of interest, rectangles, circles, etc. We develop algorithms that are capable of detecting those shapes. Since the datasets have always different qualities, thus the geometries of objects represented in the point clouds are often different from the origin. To yield more accurate results in geometry detection, the algorithms must adapt to various characteristics of data such as noise, density and so on. Each algorithm has parameters that can be set to adapt with the nature of datasets.

Group IV: "Measurement" consists of methods that allow to measure dimensions. For example, algorithms measure the dimensions (height, width, length) of a volume of subset point clouds. Angle calculation algorithms measure angle between two lines or angle between two planes based their normal vectors. Some algorithms calculate the length of a line segment and number of lines, etc. We also include classification algorithms which are able to classify the geometrical structures such as isolated points, line segments or patches of finite plane, etc.

In the following section, we introduce some algorithms belonging to the defined groups. These algorithms are just a small sample of the whole toolbox or catalogue of algorithms we use in our approach. Many more have been analyzed, modeled in the knowledge base and used for object detection.

4.3.2/ DATA PREPROCESSING

Noise reduction algorithm

Laser scanners capture geometry of 3D objects in the real-world and represent their surfaces in the form of 3D point coordinates. Either laser scanner devices or measurement processes often cause two kinds of errors in the scanned data: measurement errors, which are reasons the inaccurate point coordinates, or outliers, which are points far from the true surface of objects. Several methods exist to reduce such noise and errors in point clouds (i.e. Removing outliers using a Statistical Outlier Removal filter in PCL). Typical applications of the present work are within structured man-made environments and we found it much more effictive and simpler in such cases to work in a lower dimensional 2D

space as to rid the 3D data off errors and noise.

The approach we propose applies morphological image processing to grey-level images that are created after orthogonally projecting 3D point clouds on a 2D plane (the plane is usually a ground plane or a plane perpendicular to it). Morphology methods, such as dilation and erosion, are incorporated to eliminate the isolated points in projected images. These isolated points in 2D are the projected points from outliers in 3D point cloud. Instead of reducing outliers in point clouds, we remove the isolated points (see Fig. 4.9a) by using a kernel with a changeable size for erosion and dilation operators. The projected images first are converted to binary images. The erosion operator is then applied to eliminate the isolated points and also results in reducing pixels representing details in the image. The dilation operator is thus applied to restore the details without bringing back isolated points. Depending on the quality of data, the size of a kernel can be set properly. We symbolize the kernel size as a parameter in noise reduction algorithm that is flexibly controlled by knowledge.

The projected images of point clouds on the ground plane and side planes are obtained by an orthogonal projection from 3D to 2D, coordinates of outliers in 3D are therefore obtained by a transformation from the 2D coordinates (the isolated points are removed by applying morphology) back to 3D ones. Note that, a 3D point may or may not appeared in the 2D projected images. This essentially depends on the direction of projection. Thus, if an isolated point is detected in at least one projected image, that point can be considered as an outlier in 3D point clouds. Regarding objects that are visible by looking from a certain side, this method can quickly solve an outlier removal problem through detecting and eliminating isolated points in each projected images.

4.3.3/ Segmentation

4.3.3.1/ Partitioning Point Clouds (Spatial partitioning)

One of the challenging issues in point clouds processing is that the size and the density of data often affect the performance of numerical processing algorithms. In particular, a large dataset or high density point clouds are computationally more expensive. Reducing the number of points in the dataset can speed up the algorithms. However, this diminishes details of object in the point clouds.

Our approach is to partition the dataset into subsets of non-overlapping regions containing 3D points in specific volumes (such as cubes). This process preserves the quality as well as quantity of the data, and algorithms perform in each cube where the amount of points is significantly reduced in comparison to considering the whole dataset. Moreover, this approach allows us to select the cubes of interest and the ones which have not much information, for example, number of points inside a cube is less than a given value. The size of cube is a variable parameter that can be assigned a value depending on different factors such as dimension and complexity (in structure) of the object. Particularly, in our prototype, we use a range from 0.3m to 0.6m for the size of cube for an object with an approximate height of 7m. For an object with a complex structure and containing many geometric elements, the size of cube is set to approximate 0.5m.

Due to the fact that point clouds of objects of interest have various geometry structures, the cube size in partition processing for these point clouds is different. We thus run the partitioning algorithm on several datasets (for example, Fig. 4.10a, b) to yield appropriate

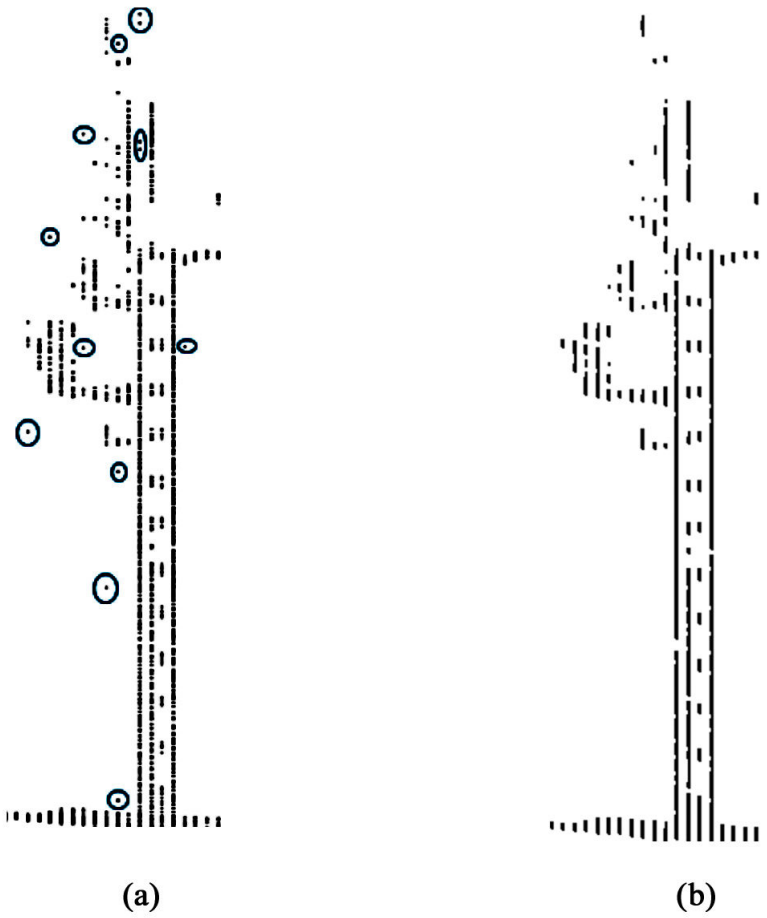


Figure 4.9: (a) Point clouds of an objects viewed from a side, with inliers represented as the points inside circles. (b) Point clouds after removed outliers by using our approach

values of the cube size regarding to particular samples.

4.3.3.2/ POINT CLOUDS SEGMENTATION

When capturing scene by laser scanners, acquired dataset (point clouds) normally contain the entire scene that includes objects of interest and others. Object detection strategies often only focus on some specific objects but not entire scene. The unused parts, which do not contain useful information, should be removed. Additionally, by eliminating these points, we significantly reduce the size of the dataset and processing performance.

Our algorithm classifies the points in the dataset based on their coordinates (x, y, z). Points whose coordinates satisfy the given conditions, such as x or y or z coordinate must be in a certain range, will be stored. In particular, there are three segmentation types:

- Segmenting point clouds region surrounding the center of an object: this case used for cropping a point cloud region (usually, a cube or a cylinder) representing the object. To determine the region, coordinates of the center and size of cube or radius of cylinder are required.

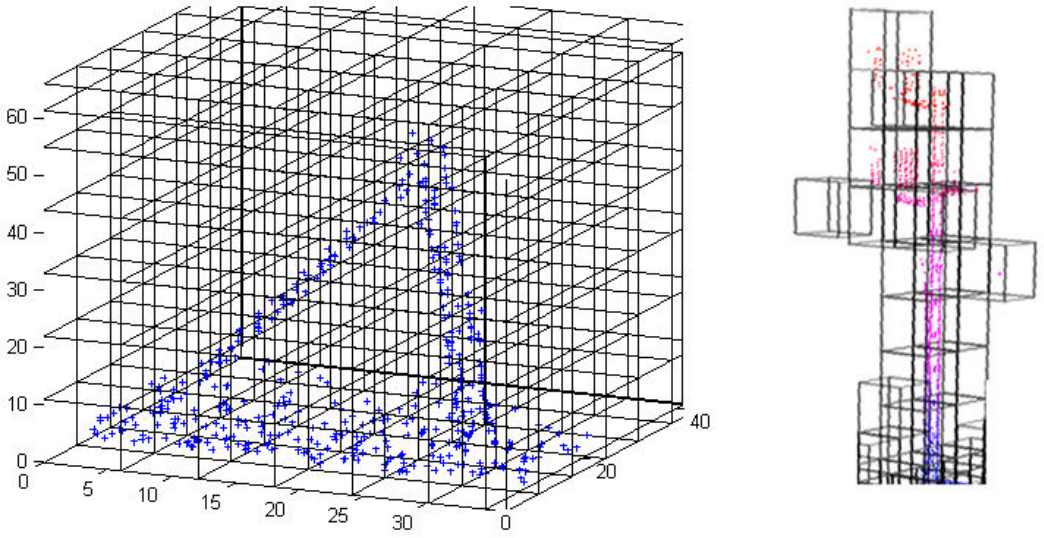


Figure 4.10: Two examples of point cloud partition

- Trimming useless points: certain conditions must be set to define the region of interest. For example, for objects standing on the ground and the scanned point clouds representing the ground can be detected. If the ground is not needed, then it is eliminated. This can be done by selecting and storing only the points whose "z" component is greater than the height of ground which is usually derived from (1) an assumption as it is approximate to zero, (2) a prior knowledge that provides the information about the height of ground, or (3) it is calculated from ground detection, such as plane detection algorithm.

We built algorithms that are able to segment point clouds in both cases mentioned above. The following examples (Fig. 4.11 and Fig. 4.12) are objects before and after processed by a removal ground point clouds. The ground information (the height) is provided from a prior knowledge.

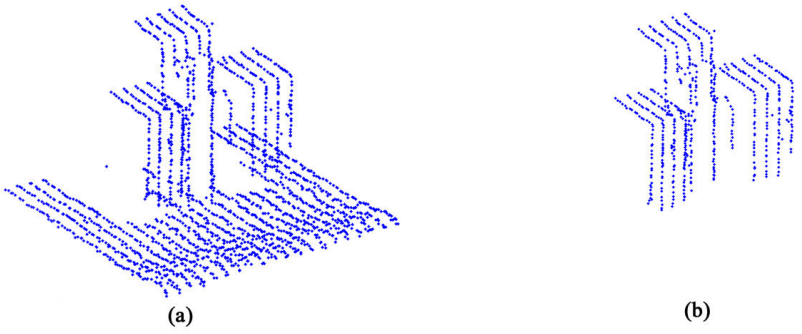


Figure 4.11: (a) Point cloud with ground points (b) Point cloud without ground points

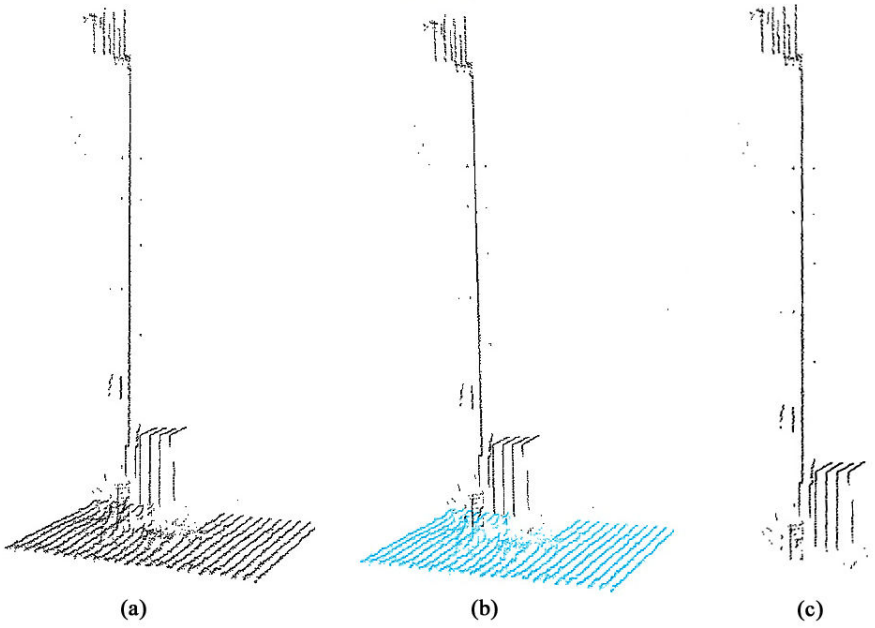


Figure 4.12: (a) Point cloud with ground points (b) Blue point clouds indicates ground area (c) Point cloud without ground points

4.3.4/ GEOMETRY DETECTION

4.3.4.1/ HULL DETECTION ALGORITHM

As far as the shape-based object detection methods are concerned, we are interested in an approach that estimates the shape of object based upon the hull or boundary of that object. The output is used to classify the type of objects that can be clearly distinguished based on their dimensions. Hull detection is applied after a noise reduction process to make sure that the detected boundary approximately represents a real shape of object. Hull detection also requires a projection from 3D point clouds to 2D image and then uses the image as an input to detect contours. From the detected contours (hull), it is possible to relatively characterize the object through its shape.

To obtain a hull, the first step is transforming 3D coordinates of points representing object of interest to 2D ones, this is simply done by omitting one component in the 3D (x, y, z) coordinate. The omitted component can be “x” or “y”, depends on direction of an orthogonal projection. The best direction is the orientation that shows the object from a side-view within a fully representation. The second step is to eliminate the projected points whose 2D coordinates are coincident. These points are obtained by projecting every 3D point on a 2D plane. This work reduces the number of points in 2D having the same coordinates.

There are two kinds of hulls: convex hulls and concave hulls. For the detection of convex hulls, an assumption is made that points in the projected image must not contain noise. This means that all points contribute to the representation of the object. The Graham Hull algorithm [Graham 1972] is applied to find out a boundary. The Graham Hull algorithm is based on sorting the points of a set around the lower point of the set by the angle that they make with the sentinel point. After sorting, the algorithm starts from the sentinel point and

computes the cross product of three successive points to find out the orientation of these points.

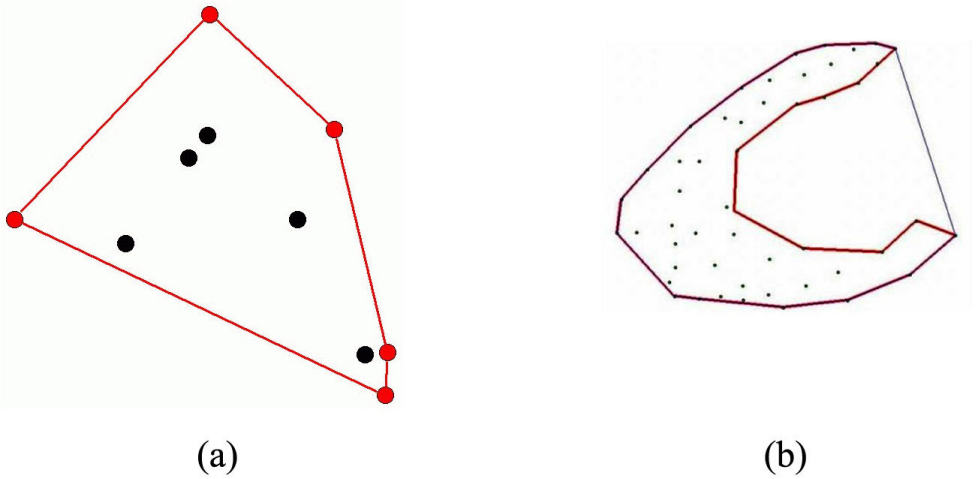


Figure 4.13: (a) Convex hull and (b) concave hull of a set of points

Regarding concave hulls, an efficient method is based on Delaunay triangulation. Generally, Delaunay triangulation starts from a convex hull and then determines, for each segment of the convex hull, if two other segments can replace the previous one principally by some distance thresholds.

A hull obtained from a set of points represents the shape of object in 2D. The hulls are used to approximately estimate the object's dimensions such as height, width and length (Fig. 4.14). Depending on the direction of the projection from 3D to 2D, which is mostly a side-view projection, we are able to obtain the two of those dimensions. The hull detection algorithm enables to classify some objects (which have distinct shapes) based on their dimensions.

4.3.4.2/ 3D TO 2D PROJECTION

Acquiring depth information of 3D objects is an important task which allows us to have rich information, such as 3D point cloud data. Such data may represent a scene like a real world but usually have a large number of 3D points. In some cases, either indoor or outdoor environments, information about location and properties of objects in the scene can be conserved on 2D representation. Particularly, when we project a point cloud onto a horizontal plane, we have a 2D image that keeps 3D points on a horizontal plane. This projection allows to decrease the size of measurement data without losing information that is needed for localization in the scene.

We project the acquired 3D point cloud data onto a horizontal plane, such as the ground plane (Fig. 4.15). Depending on the density of dataset, we can use whether an orthogonal (parallel) projection or perspective projection is appropriate. In our use case, perspective projection is applied if the density of point clouds is low such as the distance between two close points in 3D is larger than 10cm, else, an orthogonal projection is employed. This

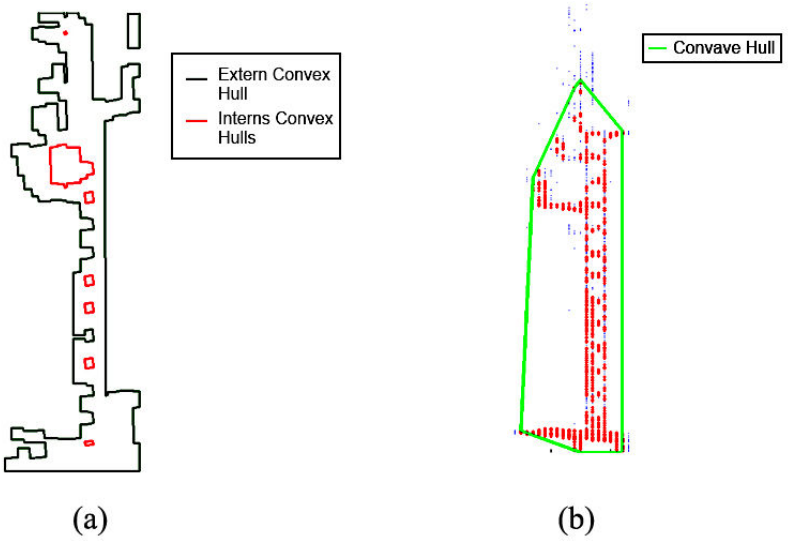


Figure 4.14: (a) Convex Hulls obtained by Morphology processing and (b) Concave Hull obtained by Graham Hull Algorithm

assumption is made to make sure that spatial information about 3D objects should be preserved on the projection image. Since objects of interest are often set on the ground and vertical on the horizontal plane, we chose to select the ground plane as a projection plane. By a 3D to 2D projection, the depth component (z) in the coordinates of a 3D point is omitted while the other components (x, y) represent the 3D points coordinate on the 2D image. Before transforming from 3D to 2D coordinate, note that all 3D points' coordinates should be normalized in such a way their root coordinates are shifted to (0, 0, 0) and all coordinates must be non-negative.

$$x'_i = x_i - x_{min} \tag{4.4}$$

$$y'_i = y_i - y_{min} \tag{4.5}$$

$$z'_i = z_i - z_{min} \tag{4.6}$$

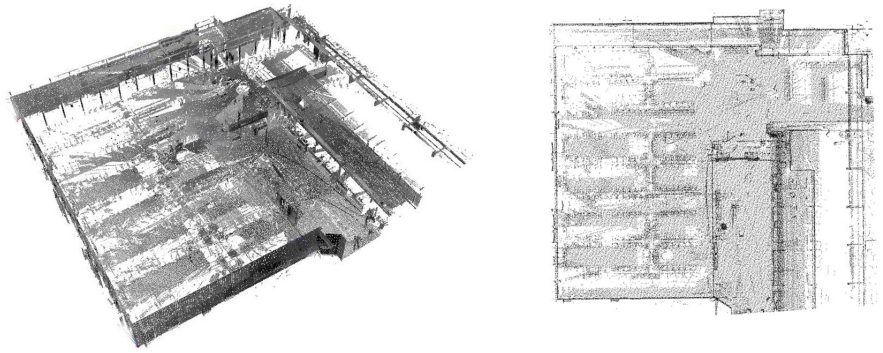
where (x'_i, y'_i, z'_i) are coordinates of 3D points after normalization and (x_min, y_min, z_min) are minimum values of the x, y and z components, respectively. The normalization is to make sure that 2D coordinates after transformation are non-negative.

Orthogonal projection

An orthogonal projection used for projecting a 3D point cloud onto a plane allows to preserve dimensions of 3D objects, except the height, on 2D plane. The transformation matrix is following:

$$P_{orthogonal} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.7}$$

In this projection, every 2D point on the projection plane has "u" and "v" coordinates which correspond to "x" and "y" in 3D. The "z" components of 3D points, which imply the depth in 3D, are absent in 2D image but would be stored. 3D points having the same (x, y) coordinates would share a same position in 2D projection image, this makes an accumulation of points and results in various intensities of 2D points. The examples below illustrate results of an orthogonal projection applied to 3D point clouds onto ground plane.



(a) (b)

Figure 4.15: (a) Point cloud of a room and (b) its projection image on the ground plane

Perspective projection

A perspective projection from 3D to 2D does not preserve the real dimensions of 3D objects. We employ this projection to scale down a scene without distorting its proportion. The scanned point cloud of the scene hold a low density in which the distance between two close points exceeds a given gap. For example, our experiment with point clouds of a railway system, typical gap between two close points is 10 cm. We therefore apply a perspective projection for this dataset to reduce the size of the scene on a 2D image as well as narrow down distribution of points on 2D plane (Fig. 4.16). A transformation matrix is expressed as following:

$$P_{perspective} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/D & 0 \end{bmatrix} \tag{4.8}$$

D in the perspective projection matrix is the distance from the projection plane to the projection reference point. New coordinates of a point in the projection image are (u, v):

$$u = \frac{D}{z}x \tag{4.9}$$

$$v = \frac{D}{z} y \tag{4.10}$$

Let us call $r = \frac{D}{z}$ the scale of the projection. Instead of choosing a value for D, we find out a proper value for r. The scale r can be determined based on several factors:

Projection image size: when the size of projection image is given, r can be calculated by $WidthProjectionImage/x_{max}$ where x_{max} is a maximum value of component x found among points in the point clouds.

Point cloud density: r is relatively calculated based on this factor. The target is to drive the projection image to appropriately represent the details of a 3D scene on a 2D plane. Therefore, the gap between two close points in 3D, due to low data density, should be narrowed down to make two corresponding pixels close together within a certain gap. The gap is representative of the dimensions for 3D and 2D. The scale r is therefore calculated by $gap2D/gap3D$.

In our example, we derive the scale based on "point cloud density" factor. The gap of two close points in 3D is 10 cm and we assume that the gap of two close pixels in 2D is 0.01cm. The scale is thus equal to 0.001.

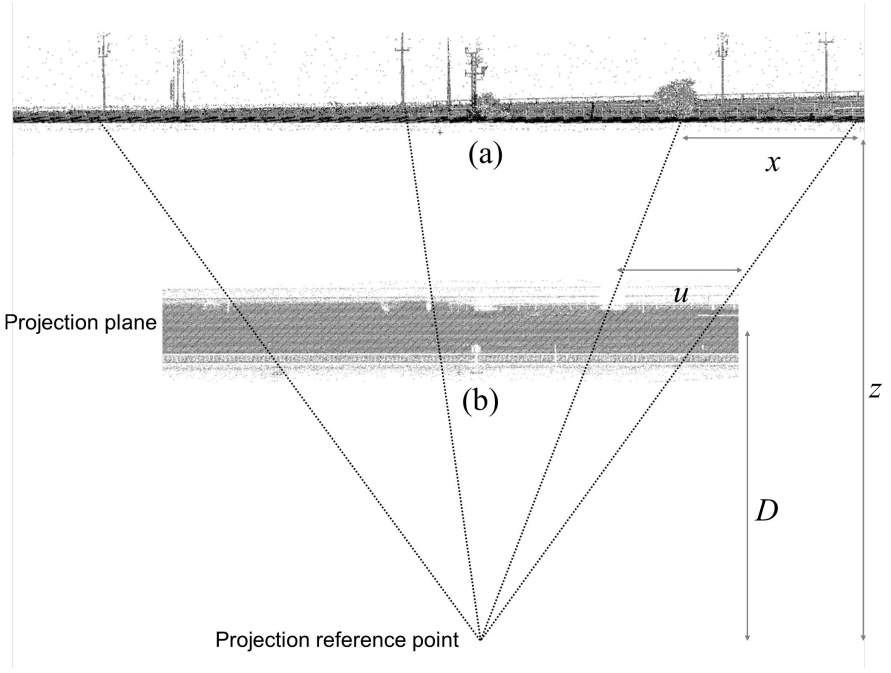


Figure 4.16: (a) Point cloud of a railroad segment, (b) a projection image of the railroad's point cloud on ground plane. The used scale is 0.001, i.e. the projection scales the scene down 1000 times. The projection image keeps all the details of the original scene from 3D.

4.3.4.3/ POSITION DETECTION

One of the important tasks before identifying objects in the scene is to detect their positions. In fact, this task is rather difficult as the difference between objects of interest

and background is regularly slight. Additionally, many kinds of object which have dissimilar shapes are present in the scene. Therefore, in order to detect potential positions of objects, our method uses the selected features to distinguish objects and background. Following this approach, we seek distinct properties of the objects thus exposing their particular locations.

All 3D objects usually stand on the ground and have different heights from the ground. This characteristic enables us to classify the ground and objects. In particular, we assumed that the reference coordinate is set on the ground, the height of objects are then often greater than zero while the ground level is approximately zero. A threshold of high level is required to classify ground and objects separately and value of this threshold depends on particular scene. Note that the sought objects may include noise or other objects which are not of interest. We employ a 3D to 2D projection to project the point clouds of the scene onto a horizontal plane that is also equivalent to the ground plane. Depending on the density or dimension of the point clouds, an orthogonal projection or a perspective projection is carried out. The 2D projection image represents the whole scene in a form of gray image which has pixels with various intensities. After projecting, the 3D points that share the same (x, y) coordinates are accumulated at the same location and increase gray value at that pixel on 2D image. Normally, an object with a certain height will appears as high density region on the image (Fig. 4.17).

Based on the information in the projection image, we apply some assumptions to be able to recognize possible positions of objects of interest:

The point clouds are points on the surfaces of scanned object. The ground is considered as a horizontal planar surface. Thus, the image of the ground on the 2D projection image is the region represented by low intensity pixels. We use a predefined threshold h_1 to classify the intensity levels.

On the contrary, high density pixels correspond to the 3D points that have the same (x, y) coordinates. Those 3D points usually are on the surface of vertical structures. Objects of interest often have vertical structures.

The "z" component (in a 3D point coordinate) has a relationship to the altitude of a point in 3D. We assume that the height of the ground is zero and h0 is a threshold. Objects of interest O_i are all above the ground and have a certain height. Then, the O_i contains the 3D points whose "z" values should be greater than h_0.

Relying on the assumptions above, we consider the possible positions of object are pixels whose intensity is greater than h_1 and pixels whose corresponding "z" value is greater than h_0.

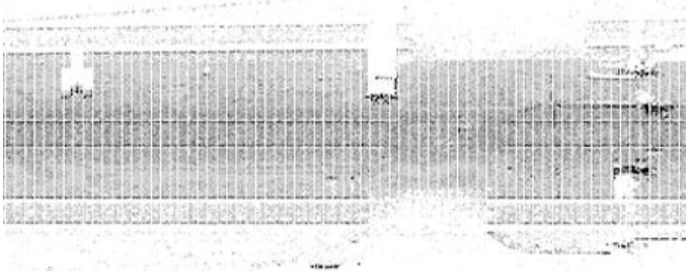


Figure 4.17: Projection image of railroad segment point cloud on the ground plane

The potential positions, the highlighted points, do not precisely locate the positions of the

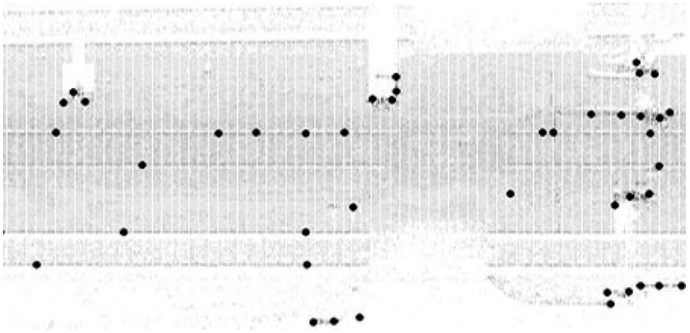


Figure 4.18: Highlighted points are possible positions of object in 3D.

objects. They may consist of noise or some footprint parts of objects. We then refine the number of the highlighted points by clustering them based on K-means algorithm. Two points having an Euclidean distance not exceeding a threshold q, are classified into a group. Value of q is equal to the maximum size of the objects (width or length). This is to make sure that all of highlighted points belong to the area. Objects should be represented by only one point. We store information about objects in the knowledge base of the system. There, their largest size is easily selected.

After clustering, each group and its center stand for a potential position of an object. By projecting back these positions to 3D for obtaining the corresponding 3D coordinates, we are able to find the object positions in the point clouds and then segment subsets of points surrounding the found positions. A subset of a point cloud, which may contain an object, consists of points not far from the center of subset (at distance q). In the next step, the system focuses on each subset to detect objects instead of considering the entire point cloud.

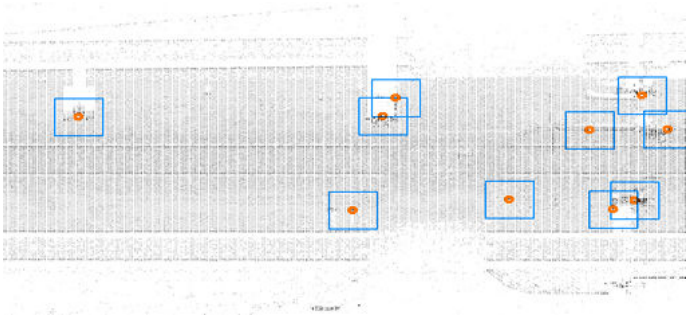


Figure 4.19: Rectangle representing potential positions of objects in projection image

4.3.4.4/ PARTICULAR FEATURES EXTRACTION

Although a 3D to 2D projection results in losing spatial features of 3D objects shown in the point cloud data, the projection image still carries some prominent features presenting the 2D shapes of the objects. Basically, these 2D shapes are lines, rectangles, circles or arbitrary shapes. Our approach is to use geometry detection algorithm in 2D to detect these shapes in the projection image, we then base on the detected 2D geometries to infer corresponding 3D geometries.

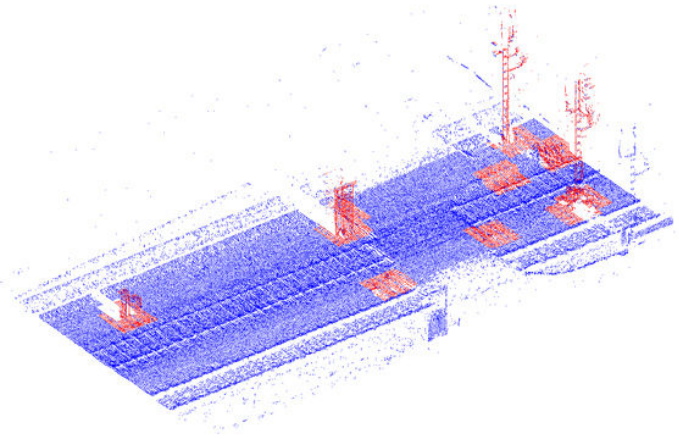
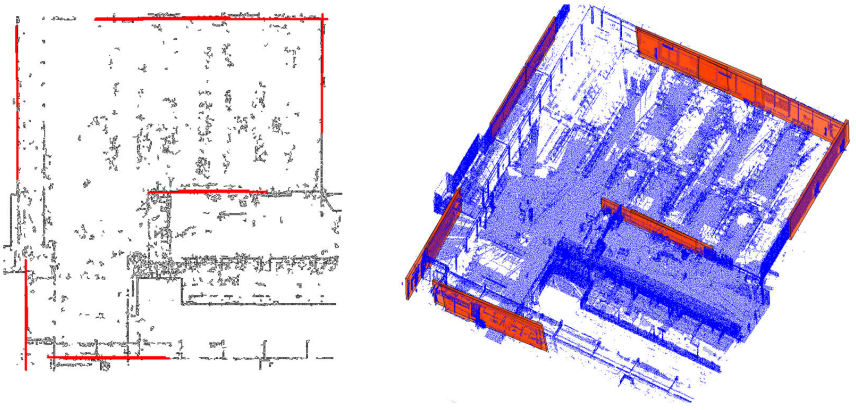


Figure 4.20: Subsets of point clouds probably contains objects in 3D

In the projection image, our method focuses on detecting 2D lines as key features to infer a vertical structure in 3D. Indeed, such vertical planes in 3D usually create linear footprints on a plane when projected onto a horizontal plane. We apply edge detection on the projection image and then employ the Hough transform technique to extract line segments from this image. Using Hough transform method enables to find best-fitting straight lines with respect to a set of points. This method also allows us to set thresholds, such as minimum/maximum length of line segments and min/max number of points used to fit a line.

We model this algorithm, with changeable thresholds, to achieve the desired lines. In our examples, we detect lines which are footprints of vertical structures in 3D such as walls, separation panels, bridges, curbs, etc... When line detection algorithms are executed, the thresholds are assigned appropriate values which are provided by the knowledge base.



(a) (b)

Figure 4.21: Vertical planes inferred from 2D extracted lines in 2D projection image. These results are obtained by using a line detection algorithm based on Hough transform method.

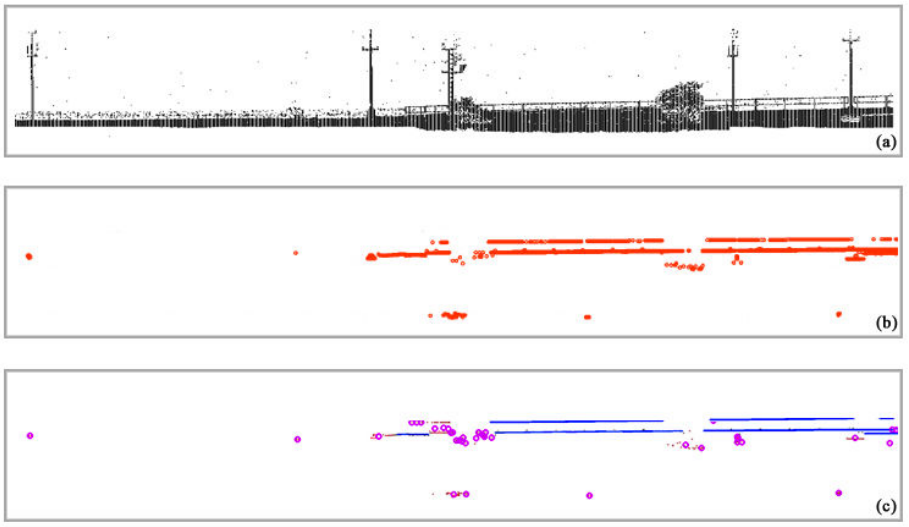


Figure 4.22: By projecting the point clouds data on a horizontal plane, we detect high intensity pixels on the image and fit these pixels to a line by using Hough transform method. The pixels fitting a line are usually vertical structures in 3D (i.e. curbs). The detected lines in conjunction with the given information about limited length of curbs, predefined in knowledge base, to uncover the curbs position in the railroad system.

4.3.4.5/ 3D LINE DETECTION

The features that are generally present in most man-made objects are surfaces and intersections between them. These features describe the particular characteristics of an object. A dataset, acquired by a laser scanner, captures the object's surfaces and represents them by point clouds. The points in the intersection between planar surfaces create linear edges which are framed as lines in 3D. They play a role as primitive shapes in recognizing the object. To extract the lines, we rely on well-known algorithms such as Least squares fitting and RANSAC to model some line detection methods.

The approach employs data partitioning techniques in conjunction with the least-squares fitting algorithm and RANSAC to fit 3D points to a line. Since the scanned data have many regions with different densities of points, the objects in the point cloud may not be fully represented. For instance, there are missing parts of object whose surfaces have insufficient points or an inaccurate representation (caused by measurement errors). Data partitioning is carried out first in the line detection process to divide data into subsets. The reasons for using data partitioning methods are as follows:

to reduce the complexity of data as well as the geometry structure of objects when data is separated into subsets. Additionally, fitting points to shapes usually yields more accurate results with small amounts of points. Subsets therefore definitely provide more reliable results than original data,

to allow the algorithm to focus on each subset. This can reduce the influence of noise as well as that of outliers. This may create errors when fitting the whole data to a model. In particular, when a subset contains noise or outliers, the line fitting errors only occur inside a subset without affecting the whole dataset.

Next, each subset processed: the least-squares method for line fitting and RANSAC

algorithms are carried out. The least squares-method is used to detect 3D line segments in every subset and then RANSAC is applied for finding longer lines which can be fitted from the detected segments in the subsets.

The least-squares method is employed to find the best-fitting line from the 3D points inside a subset through minimizing the sum of squares residuals. A residual is the distance from an observed value (3D point clouds) to the fitted value provided by a line model (line equation). Due to the uncertainty in data, usually caused by measurement error, the fitting algorithm accepts a tolerance. To adapt with this issue, we consider a line model is as a model of cylinder whose radius can be adjusted. Regarding different data conditions, we alter the radius of the cylinder through assigning proper values which are inferred from data characteristics (density, noise, measurement errors. . .) and stored in the knowledge base. Note that, to obtain these values, we perform a manual trial-and-error process for different kinds of data.

Figure 4.23: Line segments are extracted from subsets of point cloud data based on data partitioning and least-square fitting.

The results obtained after fitting points in each subset are shown as Fig. 4.23. However, these segments are separate individuals that require further processing to connect them. Our algorithm, which based on RANSAC algorithm, searches possible straight lines which might be generated by connecting the segments. RANSAC is executed through iterations to estimate parameters of a line model from an input data. The input data is not considered as all 3D points of original point clouds but only a set of the ending points of line segments. The number of points is therefore reduced remarkably which significantly improves the computational time of the algorithm. For example, if a point cloud data including 2000 points is partitioned into 100 subsets. In each subset, we employ the least-squares method to fit a line segment. The fitted line segment has two end-points. Hence, there are 200 ending points obtained in the whole data. We use RANSAC to fit 200 end-points to a line instead of 2000 points.

Based on the same idea we used for the least-squares fitting of a line, we also consider a line model in the RANSAC algorithm as a cylinder. Following the RANSAC's principle, our method takes two random end-points to produce a line model. Along this model, points

outside the cylinder (outliers) are not interesting while the ones inside the cylinder (inliers) are taken into account. A line is considered as a best fit line if the length is within some predefined range, the number of inliers is greater than a given threshold (minimum inliers). The maximum and minimum length of a line, minimum inliers and radius of the cylinder can be set through different parameters. These parameters are altered to adapt with different characteristics of the data and geometric features of an object. After the current best fitting line is found, the next iteration processes the points that are not included in the inliers in the detected line. The process ends when the number of remaining points decreases to a certain amount.

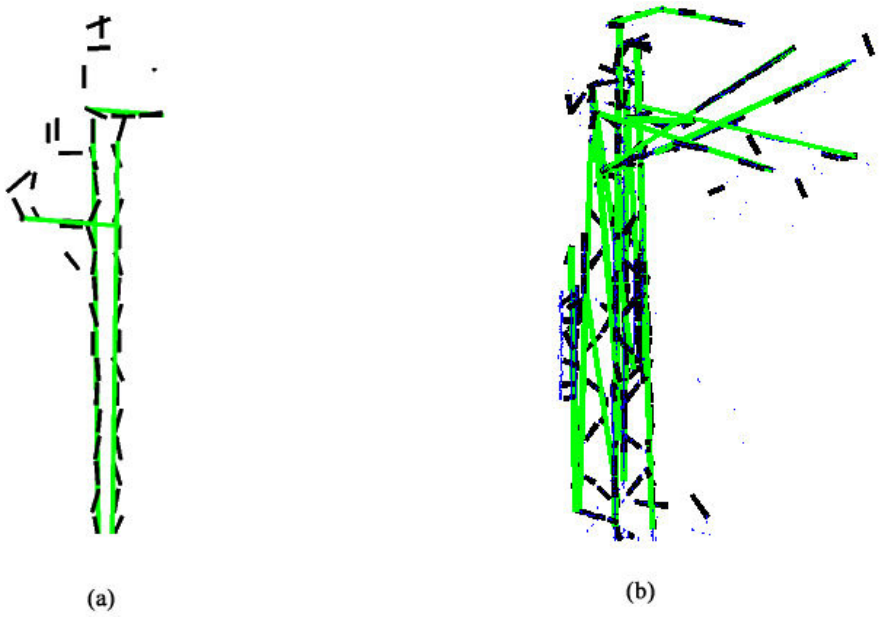


Figure 4.24: Results of extracted lines obtained by using RANSAC integrated with least-squares fitting and data partitioning

4.3.4.6/ PLANE DETECTION ALGORITHM

Laser scanners capture real-world objects and create point clouds describing the surface of the objects while some acquisition devices are capable of providing color information. However, the coordinates of the 3D points on the object's surface are mainly used as primary information. Shapes of object are diverse and basically presented by different surfaces in term of orientations, characteristics (such as flat or curved), size, etc. Among these features, planar surfaces are often encountered in man-made objects. Our approach thus focuses on this particular feature and exploits it in object detection. In the scanned data, a planar surface contains sets of points on a finite plane each possibly having different density. We introduce our plane detection algorithms to extract planes from point cloud dataset within various conditions (density, noise).

The first approach of plane detection is based on data partitioning and least-squares fitting. We proceed by partitioning the data into subsets (cubes) where small details of objects are accessible. We employ the least-squares algorithm to find the best fitting plane from the 3D points in a subset. The maximum distance from inliers to the plane

model is a threshold which can be adjusted. The detected plane is represented as a planar patch within a cube and identified by a normal vector indicating the orientation of the plane. Patches sharing the same orientation (with some tolerance) will be grouped in a class (in a distinct color as shown in Fig. 4.25) using the mean-shift algorithm [Comaniciu and Meer 2002]. The mean-shift algorithm is a nonparametric clustering technique which does not require prior knowledge of the number of classes.

In each class, we merge two close patches based on the Euclidean distance from two centers of patches. In particular, if this distance is smaller than the length of edge of a cube then two patches are connected. Consequently, we obtain patches representing the 3D points on the planar surfaces of the object. Note that, the size of a cube in the data partitioning algorithm as well as the distance threshold in the least-squares algorithm will be altered depending on the each situation.

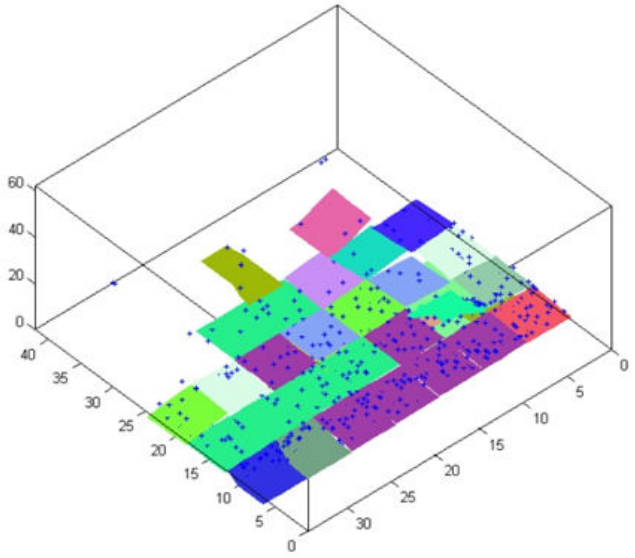


Figure 4.25: Point clouds are partitioned into subsets. By using the least-squares algorithm, we extract planar patches from 3D points inside the subsets. The same color patches illustrate co-planar planes.

The second approach uses RANSAC to detect planes from point cloud data. RANSAC enables to extract the best fitting plane after the first iteration. The points in the detected plane will be used in the second iteration of the fitting process. The process stalls until the number of remaining points reaches a threshold (in our use case, this threshold is not smaller than 20% total number of points in the original data).

4.3.5/ MEASUREMENT

The group of “Measurement” algorithms contains calculation methods to measure three dimensions (width, length and height) of an object represented in a point cloud. The aim of this algorithm is to determine and calculate the dimensions of a specific region that includes an object of interest. In fact, point clouds do not only represent a region of the object but also noise and outliers caused by measurement that need to be accounted for. To expect a precise approximation, our method reduces the influences of uncertain data

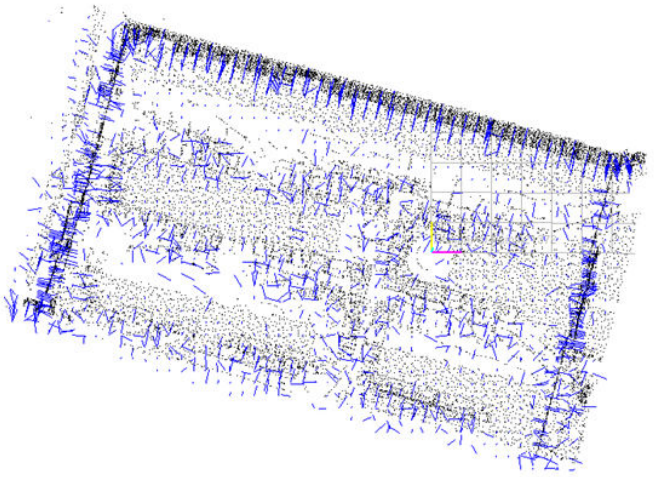


Figure 4.26: A point cloud of a room is partitioned into subsets. By using the least-squares algorithm, we extract a plane in each subset and the normal vector of the plane is therefore determined. The figure illustrates normal vectors with different orientations.

in calculating dimensions of an object in a point cloud.

We focus here only on the height of object as the width and length can also be calculated in similar way. Basically, the height of an object is determined by the distance from a lowest point Pbottom (usually ground level) to a highest point Ptop within a point cloud representing the object. Finding the highest point is rather a difficult task. Indeed, a scanned point cloud data often contains outliers an noisy points that may be present around the Ptop. Our method is to give an approximate value for Ptop through calculating an average height of highest points – which have almost the same level to the real height of object. We select a given number of surrounding 3D points that have maximum "z" component values and then calculate their average. The height of an object is approximated as a distance from ground level to the mean height of the selected points. Point cloud data usually have different quality, thus to obtain an optimal number of selected points (having highest "z" values), we must manually deal with each. For that, we carry out some experiments in our dataset. The number of selected points is in a range from five (if data has less noise) to ten (if data has much noise). This allows us to get reliable results of height calculation.

This method can also be applied for calculating width/length of an object as points have minimum and maximum "x"/"y" value will be taken into account. Width/length of an object is approximated by a distance from minimum "x"/ "y" to maximum "x"/ "y" values.

4.4/ Algorithm Selection Module (ASM)

The ASM contains multiple algorithms to solve specific tasks such as point cloud segmentation, model fitting or feature extraction. First, the ASM connects the algorithms as a graph. Next, the ASM selects suitable algorithms and connects them (as a sequence) for processing in specific situations. Finally, algorithm parameters are set to proper values in order to increase their performance.

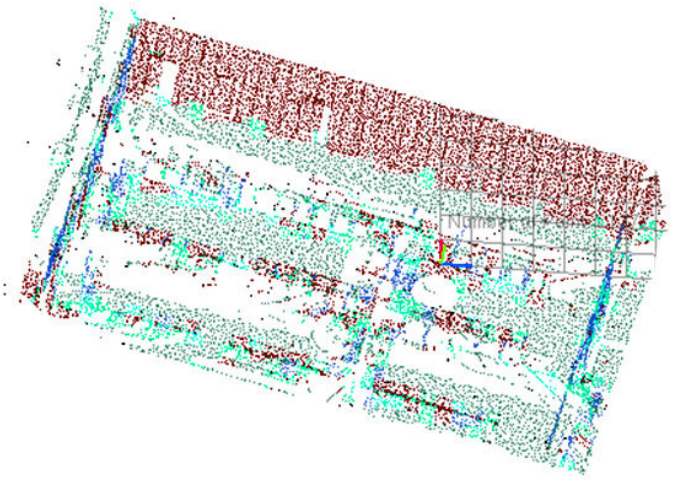


Figure 4.27: After classifying normal vector of planes based on their orientation, the result shows the 3D points that belong to the plane are colored by a distinct color. Some objects such as walls and floor can be detected.

4.4.1/ Modeling algorithm in knowledge base

The entire algorithm library is also represented - in another part of the ontology - in a way that is similar to the representation of objects. For instance, we model the attributes of our algorithms and the relationships that may exist between them in a class named “Algorithm”. Based on their general purposes, the algorithms are divided into groups: “Data Processing”, “Geometry detection”, “Segmentation” and “Measurement” (Fig. 4.30).

The more specific purpose of an algorithm is defined by the field “isDesignedFor” in the ontology. By this attribute, we can link an algorithm to a relevant property of an object and therefore create relationships between algorithms and objects. The input data type of an algorithm is determined through “hasInput”. Similarly, “hasOutput” defines the output data type of an algorithm after processing. Each algorithm is also defined by its predecessor through “hasPrerequisite”. The predecessor is an algorithm that needs to be executed before the algorithm under consideration. By collecting such information, we build a concrete model for an algorithm as: AlgorithmName {hasInput, hasOutput, hasPrerequisite, isDesignedFor} (Fig. 4.31).

4.4.2/ Algorithm graph

The algorithm characteristics and their possible inter-relationships were modeled in AlgorithmKnowledge. All possible connections among algorithms are represented through a directed graph, in which nodes are algorithms and edges are the connections between them. We define three algorithm properties in the ontology: “hasInput” as input data type, “hasOutput” as the output data type after processing and “hasPrerequisite” implies to appropriate algorithms should be executed first. Note that, detecting a particular feature of an object generally requires a processing sequence. The relationships are defined through compatible “hasOutput”/“hasInput” attributes. Since the position of each algorithm in a sequence needs to be respected, the “hasPrerequisite” attribute must also be set to the appropriate groups of algorithms For example, the output from “Color Image Im-

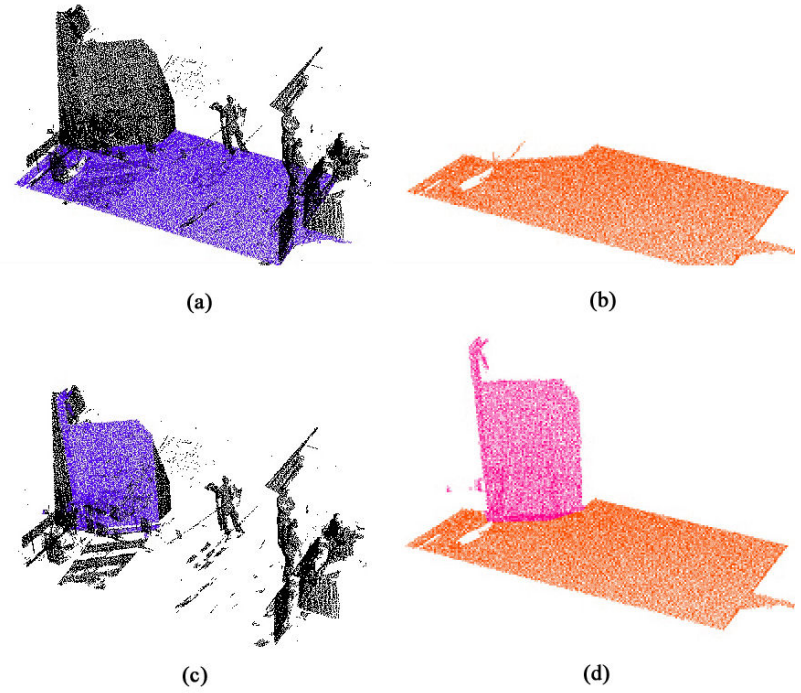


Figure 4.28: Planes extracted by RANSAC algorithm

port" algorithm is compatible with algorithms having a "color image" as input, such as the "Color Normalization." In the knowledge base, this information is represented as "Import-ColorImage" has "hasOutput" = "color image" and "ColorNormalization" has "hasInput" = "color image". By collecting such information, a concrete model for an algorithm is built with the following structure: AlgorithmName {hasInput, hasOutput, isDesignedFor}. All connections in the graph must be taken into account in order to extract an algorithm sequence as a path from one algorithm to another. Djikstra's algorithm was used to find the shortest path from the starting algorithm to the desired one. This approach prevents the algorithm sequence from forming an endless loop and results in the acquisition of an appropriate sequence.

The example shows a graph with directed connections between pairs of algorithms in a set. The algorithms have been denoted A, B, C, D, E, F, G, H (Fig. 4.32). The connections are established based on "hasOutput"/"hasInput" and "hasPrerequisite" criteria.

4.4.3/ ALGORITHM SEQUENCE EXTRACTION

The input point cloud contains objects to be identified. The identification task is made more difficult and challenging in the presence of noise and/or occlusions. The ontology schema describes the scene through categories of objects that might exist in the scene, their characteristics and their relationships to each other. The scene thus comprises different objects with large number of properties and relationships. The impact of object related knowledge is not restricted to the classification alone as it also affects the algorithmic processing. The selection and behavior of algorithms are not independent from other factors such as the type and characteristics of objects and data. Different algorithms are designed for different contexts. These differences can be addressed and

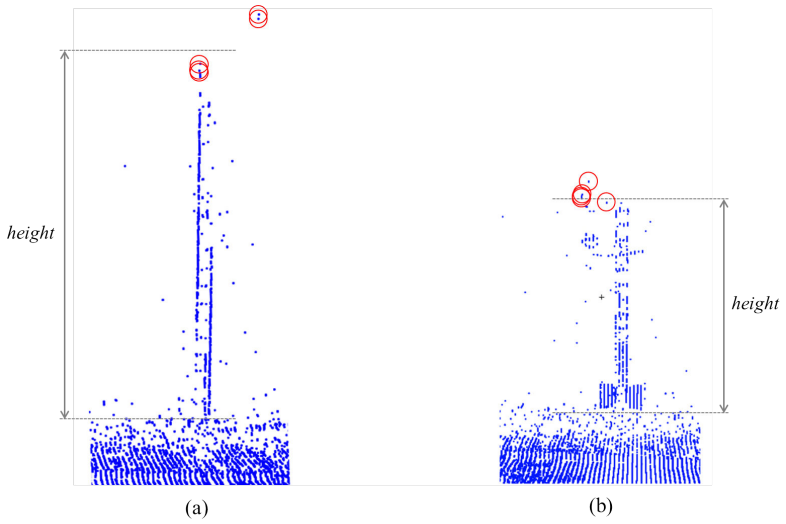


Figure 4.29: An example of height approximation, height is counted from ground to the mean height of five selected points on the top.

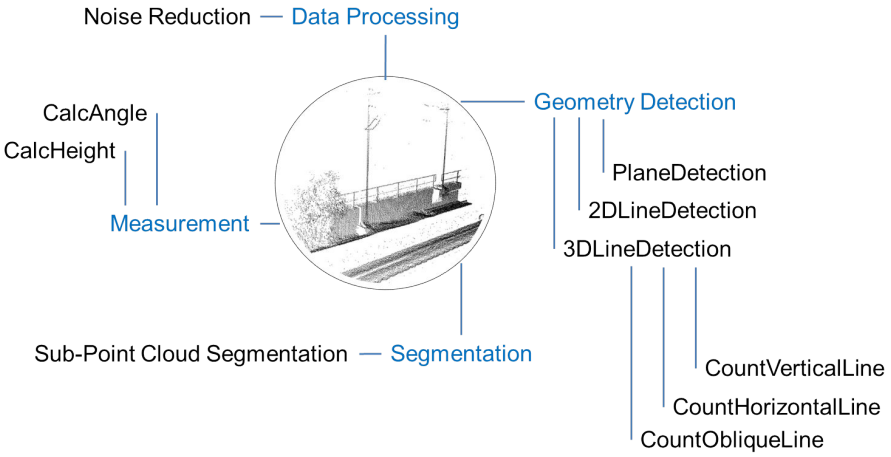


Figure 4.30: Algorithm constitution

properly modeled. For that purpose, the knowledge base hosts the algorithm knowledge which is linked to other classes inside the knowledge, such as scene knowledge and data knowledge. This allows for the modification of the usage (e.g. parameter, sequences) of algorithms corresponding to the knowledge base details.

The ASM generates a sequence of algorithms, based on two conditions:

1. Algorithms are selected for a given object property,
2. Connections are established based on the possible routes in the algorithm graph.

For the first condition, ASM seeks algorithms having their *"isDesignedFor"* value that matches the property under consideration. Some object properties can be detected by not only one but also several other algorithms leading to various possible sequences. The second condition, leads to several links between the algorithms following the similarity input-output criterion as well as the order of processing. In order to extract an algorithm

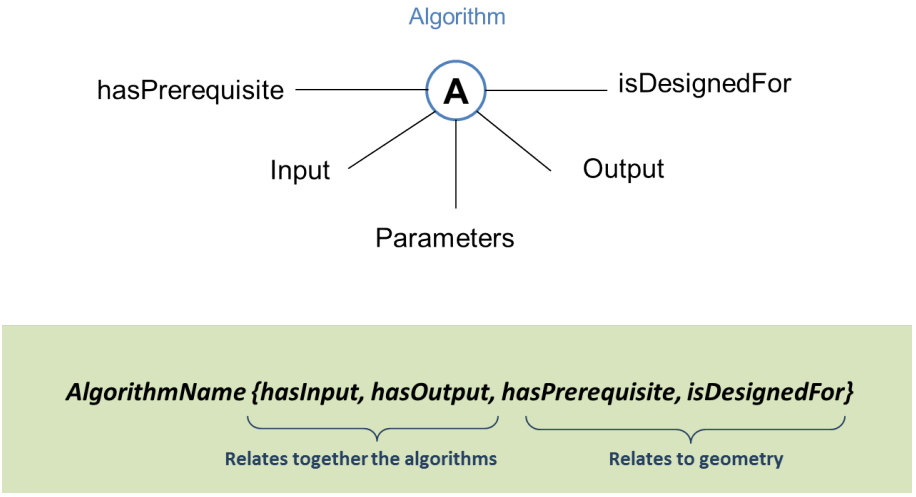


Figure 4.31: Model of an algorithm

sequence as a path from an algorithm to another, all connections must be taken into account. In general, there will be more than one path from one algorithm to another. In the graph example, if we need to execute D starting from A, there will be three possible paths: $\{A, B, F, D\}, \{A, B, E, D\}$ and $\{A, C, E, D\}$. Each edge of the resulting graph of algorithms is then assigned a weight based on algorithmic properties such as: processing time, result quality and memory consumption. The weight "w" of each edge depends on which algorithmic property should be considered in priority. It is controlled by three values: α, β, and γ such as:

$$w = \alpha.Time + \beta.Quality + \gamma.Consumption \tag{4.11}$$

in which "Time", "Quality" and "Consumption" are measured in a normalized frame. In the present state of our system, the values of α, β, and γ - which measure the importance one wants to assign to each of the three criteria ("Time", "Quality" and "Consumption") - are left to the discretion of the user. In order to choose the appropriate algorithm sequence, we use the well-known Djikstra's algorithm ([Dijkstra 1959]) for finding the single shortest path in the graph leading to the desired algorithm (Fig. 4.33). This approach has the advantage of preventing the sequence of algorithms to form an endless loop and allows for finding an appropriate sequence.

Once an algorithm sequence is executed to detect an object feature, each algorithm in the sequence should be configured with proper settings. In brief, relying upon the knowledge of the data and scene, algorithms are altered to adapt with different characteristics of data as well as objects in particular situations. This technique is introduced in following section.

4.4.4/ KNOWLEDGE-BASED ALGORITHM CONFIGURATION

The system is modeled as a knowledge-based configuration whose algorithms perform differently under various conditions. The algorithm efficiency depends on multiple factors which could include the geometry characteristics, data properties, and viewing angles. Parameter modification is mostly done manually through a number of simulations, which

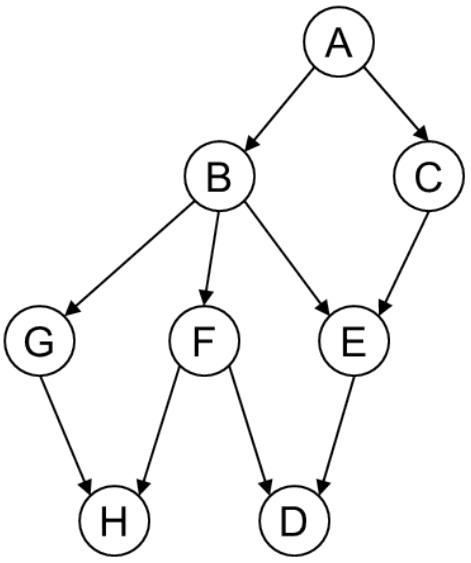


Figure 4.32: Vertices are algorithms and edges are connections between them.

execute the algorithm against different settings and then evaluate the best parameters therein.

Objects of interest are represented by the class Domain concept, consisting of geometries which are stored within the Geometry class. These geometries have certain characteristics, such as thick, thin, flat, or broken. The DataKnowledge contains information about datasets (point clouds, images), and provides characteristics such as low density, high density, distinct, invisible, and noise, which can also influence the algorithm selection. Both geometry characteristics and datasets are collected in a Characteristic class. Algorithm knowledge is coupled with scene knowledge for appropriate algorithm selection, for example, the system automatically invokes “3DLineDetection” to detect line features in point clouds and “2DLineDetection” for the same line features within an image.

We implemented a “trial-and-error” simulation in order to obtain the values of an algorithm’s parameters under different dataset states and object characteristics. For example, the thick and thin characteristics are related to the instances of RiskBenefit. After a “trial-and-error” simulation is done by an expert, a “LineDetection” algorithm with a threshold value of 0.08 m (a threshold in linear model fitting by RANSAC), for example, could extract a thick line while 0.03 m would be suitable to extract thin lines. All the instances and corresponding algorithm parameters (including the threshold values) are then stored within the algorithm knowledge.

The following example explains in detail how an algorithm can be selected based on the given conditions. Different numerical processing algorithms are executed to detect the objects in the scene represented by class Domain Concept. The object classes in the Domain Concept constitutes geometries that are stored within class Geometry. These geometries have certain characteristics *like thick, thin, flat, broken, etc*. These characteristics are present in the class Characteristic. These characteristics present first impressions of risk or benefit that the algorithms can use. Different data sets (i.e. point clouds, images) provide some characteristics that can influence the choice of algorithms or at least its parameters. For simplicity of the present demonstration example, we use different data sets, a 3D point cloud and an image, whose characteristics have an influence

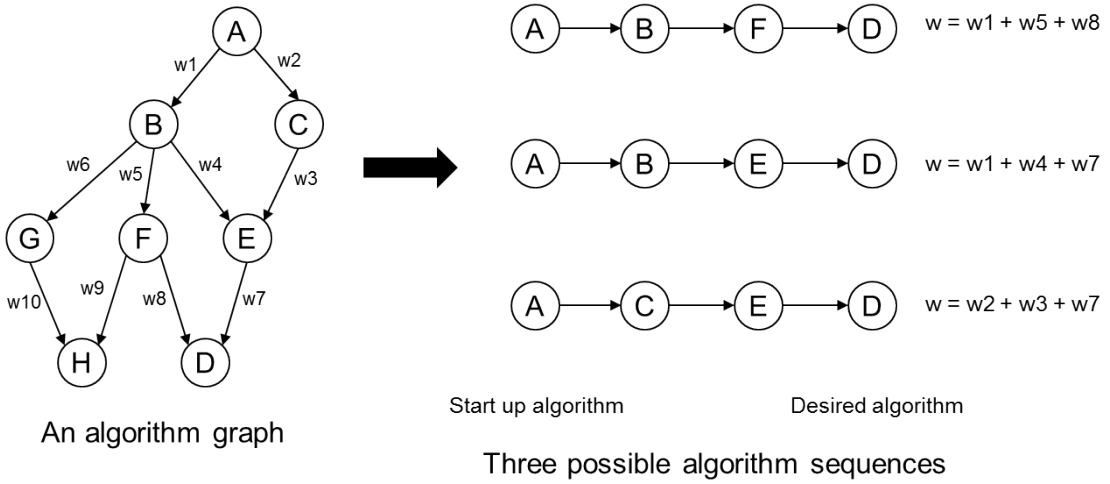


Figure 4.33: Algorithm sequences extracted from the graph. The sequence with minimal weight “w” will be selected.

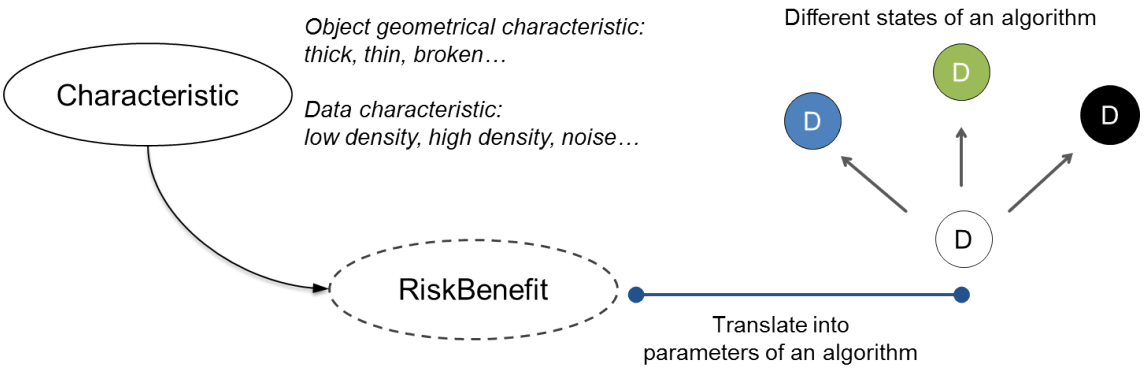


Figure 4.34: The influence from instances in RiskBenefit on an algorithm

on the algorithms. We use two algorithms in this example: *RANSAC-based Line Fitting (RLF)* and *Line Detection from 2D Hough Transformation (LDHT)*. Both of them detect lines. However RLF is executed in 3D point clouds while LDHT works on images. This distinction should be reflected in the algorithm knowledge. The conditions that LDHT and RLF are designed for line detection and work on 2D and 3D environments are represented through description logic axioms:

$$LDHT \equiv \exists isDesignedFor.Line \tag{4.12}$$

$$LDHT \equiv \exists bestSuitedFor.\{2D\} \tag{4.13}$$

$$RLF \equiv \exists isDesignedFor.Line \tag{4.14}$$

$$RLF \equiv \exists bestSuitedFor.\{3D\} \tag{4.15}$$

Regarding the 3D environment, we simulate RLF in different settings to determine what parameters need to be set at which setting. For simplicity, we assume that there are two characteristics of line: thick line and thin line are concerned. To detect two kinds of lines, RLF should be altered (by changing parameters) to adapt with each case. In short, there are two instances within the class RLF that represents these two broader distinctions: *RLFDistinct* and *RLFIllusive*. *RLFDistinct* represents suitability to the distinct lines which has 0.08 as threshold value (a threshold needed in RANSAC algorithm) whereas *RLFIllusive* represents the suitability to illusive lines having a 0.05 threshold value. If the line is *thick* then it prompts *RLFDistinct* and if *thin* it prompts *RLFIllusive* within class RiskBenefit. These are presented through description logic axioms:

$$\langle RLFDistinct, Distinct \rangle \in bestSuitedFor \tag{4.16}$$

$$\langle RLFDistinct, 0.08 \rangle \in thrs \tag{4.17}$$

$$\langle RLFIllusive, Distinct \rangle \in bestSuitedFor \tag{4.18}$$

$$\langle RLFIllusive, 0.05 \rangle \in thrs \tag{4.19}$$

The definitions of the geometries of the objects in the class Domain Concept are used by the algorithms to determine the parameters. In an example of detecting a *Signal* in a railroad system. *Signal* is an object of interest defined in the class Domain Concept which is constructed by vertical and horizontal linear elements (lines). In the point cloud data, a *Signal* appears as a clearly visible object with thickness of lines. The *Signal* should be modeled in the knowledge base accordingly.

$$Signal \equiv \exists hasDataNature.PointCloud \tag{4.20}$$

$$PointCloud \equiv \exists hasProvision.\{3D\} \tag{4.21}$$

$$Signal \equiv \exists(hasGeometry.Line)AND(hasCharacteristics.\{thick\}) \tag{4.22}$$

The restriction axioms presented in equations 4.20 and 4.21 provide the information that the algorithms for *Signal* detection are designed for 3D environments. Likewise, the axiom presented in 4.22 provides a clue that the *Signal* has thick lines. If we go to equation 4.14 and 4.15, we can reason that the algorithm RLF is best suited to detect lines in the Signal. Now we come to which parameter should be passed to RLF to detect thick lines. The *thick* and *thin* characteristics are related to the instances of RiskBenefit which is presented by axioms:

$$\langle Thick, Distinct \rangle \in hasProvision \tag{4.23}$$

$$\langle Thin, Illusive \rangle \in hasProvision \tag{4.24}$$

Putting together axioms from equation 4.23 and 4.16, the knowledge base can reason that the best choice for line detection of the *Signal* in the railroad scene is instance *RLFDistinct* of class RLF with threshold value of 0.08.

Through this simple example, we have seen that with the modeling the knowledge pattern of the algorithmic execution, one can make the algorithmic processing intelligent. We have also witnessed the manual learning mechanism through simulating the algorithmic execution in different settings could be transferred to the knowledge base where the machine helps in recommending the best suited algorithm for the particular case. This mechanism however needs to be tested in other settings to make it even more comprehensive.

4.5/ Integration of knowledge into 3D processing

In this section, we introduce object detection strategies and the concept of integrating knowledge into 3D processing. The detection strategy depends on the availability of knowledge from various sources and how to model knowledge for different purposes. Regarding particular detection strategies, the acquired knowledge is stored and organized in a specific way in order to be accessible for the reasoning process. We first present knowledge-driven strategy in general. Then, two approaches are introduced as particular methods of using knowledge to manage the process of object detection.

4.5.1/ Knowledge-driven strategy

The knowledge formalization is based on the understanding of underlying semantics and processing it through technologies such as OWL. The top-level ontology presents the main knowledge framework and holds generic semantics for all addressed domains. For the case studies, this contains: the scene, object geometries, spatial relations and algorithms. It originates from existing knowledge sources, such as information systems, guidelines as well as rules of the carrying institution, and an extensive study of the sample scenarios. Logically, quality and completeness of such formalized knowledge have a large impact on the quality of the results, and also have to be adapted to each individual application domain.

Such large differences in the knowledge base clearly must have impact on the guidance of the algorithms and on the strategies used. In principle, the more knowledge existing, the more precisely and directly the algorithms can be guided. There are strategically different concepts following the degree of quality for the knowledge. Hence, we distinguish between detailed knowledge case and generic knowledge case. In a simple scenario, with concrete information about potentially existing objects, for example known through CAD or Industry Foundation Classes (IFC) files, the detection strategy can be guided more easily and may be reduced to a change detection problem. In the more generic and difficult case, such a framework only contains the abstract and general knowledge of object categories, the structure of a scene, geometric relations between objects, the structure of data, the nature of algorithms and the potential relationships between all these components. In short, three major strategies are concerned:

1. Defined specific knowledge-based processing: Change detection in a scene with targeted objects are known and their positions in a point cloud are given.

2. Defined specific knowledge-based processing: Object localization with targeted objects are known and their positions in a point cloud are unknown.

3. Generic knowledge-based processing: Object detection and identification with unknown object as well as unknown position, only properties of all objects are given.

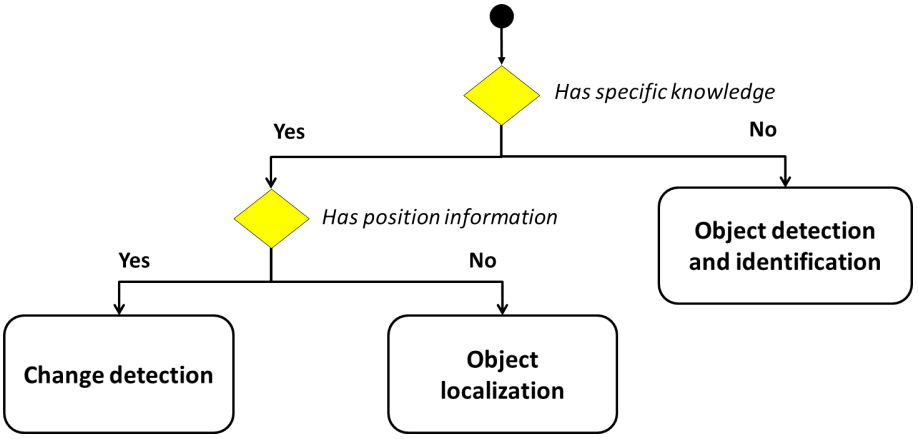


Figure 4.35: Knowledge-based object detection strategies

We proposed these strategies for each individual application case which has its own framework of knowledge. The content of such a framework changes with the domain to which an application has to be referenced (architecture, industry, civil engineering, etc.). Accordingly, knowledge models to be used must be different. In addition, the framework will be influenced by the amount of available knowledge in a particular application. This may spread a large field, starting from extensive and actual databases with more or less precise information up to just some general ideas about the objects in question and without any direct data on the other end.

4.5.2/ SPECIFIC KNOWLEDGE-BASED PROCESSING

Data sources are various. This might range from simple CAD plans over spatial information systems to object-oriented databases supporting data in rich and complex formats like document, CAD, etc. Based on these data sources, the different levels (scene, geometry, topology) in our knowledge model can be expressed as much as possible. In an ideal case, we therefore might know about the semantics of objects (there are walls, floors, ceiling, etc.), the geometry (position, extension, orientation, etc.), additional features (roughness, color, other surface characteristics) and topological relations (wall A sits on floor B). This would give an important base for a detection strategy. This knowledge is then linked to the algorithm knowledge and allows to guide the processing part.

4.5.3/ CHANGE DETECTION IN A SCENE

Official topographical data sets have an age between one and about twenty years. Even continuously evolving objects like industrial plants may have data sets of similar age. But even for early data sets an analysis could be of interest as the contained objects may undergo permanent changes. For example, objects inside a building at an airport (where we have conducted some experiments) are often changed and moved. Obviously, in

many construction sites visible on airports all around the world, building parts, and elements of infrastructure are undergo many changes. Walls disappear, new walls show up, new openings or closings inside walls arise and elements of various technical infrastructures get modified. Normally those changes are not updated into the databases. An airport might seem as a special example, but there are many similar scenarios for aged datasets. One reason for missing updates of databases is the cost of such maintenance. In practice, update measurements are done manually and due the amount of time to be invested it may turn out to be very expensive.

The goal of our system is to verify the presence of a known object at a given position in point clouds. Change detection is a comparison between sought objects in a scanned data with a defined one in the knowledge base at the same positions in scene. Knowledge provides object positions of interest and object properties at those positions. This strategy takes all properties of the object and ASM invokes appropriate algorithms to create a sequence of algorithm to be executed. The entire results after detection, including geometries and their dimensions (i.e. 3D line coordinates, number of lines, length of lines, etc.), are populated in the ontology. By comparing these results to the defined properties of the corresponding object, we are able to conclude that an object is still in the position of interest or it has been moved or replaced by another object. This strategy enables to detect the changes in a scene before and after a certain time.

4.5.4/ Object localization

In previous cases of an available position, the system guides a detection process spatially (at a given specific position) and semantically (with known object properties). In this case, object localization knowledge provides us with the definition of the sought objects. Knowledge also describes the scene in terms of the spatial relationship between objects which tells us how the objects are scattered in a scene. The goal of object localization is to determine the positions of objects of interest in the point cloud based on mutual relationships with the other objects.

The scene knowledge is described in the schema of ontology and includes semantics of the objects not only properties, restrictions but also relationships between objects. A determined position of an object in the scene may allow an algorithm to find other positions of less prominent objects which are complex or invisible to be identified by detection algorithms. The more information about an object position and spatial relationships are determined, the more accurate the detection for other objects is. Spatial knowledge models geometric relations between either components in an object or objects in the scene in the ontology. The object localization strategy utilizes geometric relations between objects to find object positions based on the determined positions. For example, it is often seen in a room that a table has to sit on a ground floor and chairs may have close adjacency to other chairs or to tables. We assume that the floor and tables are identified while chairs are more complicated to detect by algorithms (or due to missing data at chairs' area). The system bases on the rules, which draws the spatial relationship between chairs and tables to identify the chairs' position. Note that, the scanned data might have different levels of quality. Areas with poor data quality, for example, can provide inadequate information for the detection process. In such cases, approximated positions can be found by using known relationships (defined in Spatial Knowledge) of the detected object positions.

4.5.5/ GENERIC KNOWLEDGE-BASED OBJECT DETECTION

With a low amount of knowledge, especially with lack of prior information to a position, the processing has to use a largely different strategy. In a generic knowledge-based object detection issue, the definition of objects such as properties, restrictions and spatial relations are available in the knowledge base. The task is to identify all objects present in the point clouds without knowing of what object should be found first or later. Initially, the most common property is defined in the set $\{Pc\}$ object is selected to detect and find the prominent features from objects in the scene. This allows classifying and annotating potentially detectable objects based on this property. The ASM invokes appropriate algorithms which are used to detect the selected properties, and then generate a sequence of algorithms to execute. For example, the most common property of electric poles - objects often seen in a railroad system - was found to be “vertical structure”. By projecting the point cloud following the vertical direction, we are able to detect possible object positions based on the density of points in the projection image.

The coordinates of the detected geometry properties as well as the detected objects are stored in a part of the ontology and considered as “individual”. The annotated individuals are identified as confined within bounding boxes in the sub-point cloud. There are three types of annotations that we assign for each individual at this stage:

Unknown: for individuals with detected properties but with insufficient knowledge for identification;

Ambiguous: individuals possibly having more than two labels;

Identified: individuals potentially within one label but still requiring further processing.

These individuals and their labels will be passed again to the processing side as new inputs in the next iteration. The process is repeated to update and improve the quality and accuracy of the results.

Next, the subsequent iterations focus on individual bounding boxes obtained from the initialization step rather than using the entire point cloud. Each sub-point cloud containing an individual will be processed to verify the object’s existence. Object definitions such as properties or restrictions are provided by the Scene knowledge, and detected by suitable algorithms with their appropriate parameters, provided by the ASM. Appropriate parameter values are set depending on the data quality at a specific area (noise, low density, etc.) and/or the object geometry characteristics *(thin, thick, etc.)* which are instances of RiskBenefit. In particular, if a bounding box is labeled *“Unknown”*, this means the individual has not been identified yet and requires detecting new features. ASM will take a decision of using other common properties in $\{Pc\}$ - other than the ones that have been used in the previous iterations. With an individual having *“Identified”* label as $L = \{O_k\}$; ASM generates a proper algorithm sequence to detect properties of Ok which have been defined in the ontology. *“Ambiguous”* label contains more than one object as $L = \{O_i, O_j \dots O_k\}$. In this case, the differences between objects in $\{O_i, O_j \dots O_k\}$ will be taken into account. This is done by finding the discriminatory properties of those objects. ASM extracts the algorithm sequence that is able to detect discriminatory properties. Based on the selected properties in different states of a label, suitable algorithm sequences are extracted from the graph and then executed to recognize object features. The process is repeated until all individuals have been annotated completely. The labels therefore are updated and improved continuously.

Starting from the initial situation, the process iteratively updates the knowledge base at certain stages. At the beginning of each iteration, the content of the knowledge base is used to detect new features. This may be a new object or a new component of an object. These new geometric features are passed on to the knowledge base in order to extend the knowledge base for the next step of classification. This classification is guided by the content and the structure of the knowledge base, which has reasoning capabilities, based on property restrictions or rule languages (such as SWRL) and refines the actual content. This refined content is used in the next iteration. The process is repeated until all entities have been completely annotated and meet the following convergence conditions:

1. All objects defined on the knowledge base are detected and annotated (simple change detection).

2. A predefined number of iterations without refinement for any entity have been reached.

4.6/ INTEGRATING KNOWLEDGE INTO PROCESSING TECHNIQUE

Robust and efficient data processing algorithms are generally available in the form of libraries of independent source code. An interface has to be implemented giving access to efficient programming languages, like C or C++, for example. Fortunately, Java provides all necessary structures to build such an interface and therefore acts as bridge to combine these real different worlds of semantic processing and efficient data processing.

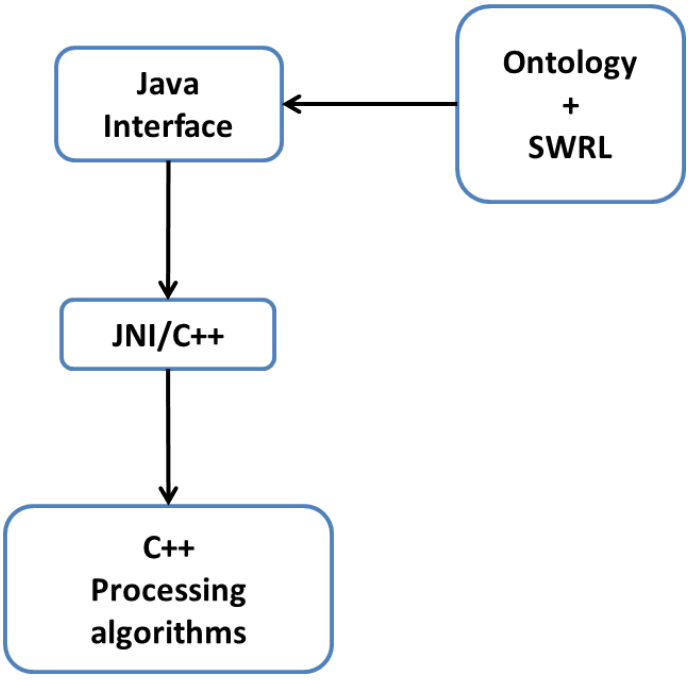


Figure 4.36: Processing architecture

As shown in Fig. 4.36, JAVA has interfaces to the semantic world and also to the processing world. As consequence, it is possible to start and control activities inside the processing environment based on functions implemented in the semantic framework.

In the simplest case, this would allow to define a C or C++ function representing a defined sequence of processing steps and to start this function as a semantic built-in method. In that case the 3D-processing wouldn't do more than just return information to be used in a further reasoning process on the knowledge level. However, this is already useful to exploit the potential of knowledge management for the guidance of 3D-processing.

However, such a solution of defining individual processing sequences and connecting them to an own spatial built-in method is of limited flexibility. It would need a large number of methods representing a complete toolbox covering most of the possible processing situations. This might result in a certain redundancy between processing built-ins for similar objects or for the same object to be analyzed under different conditions.

A higher degree of flexibility and less redundancy could be achieved by developing an separate processing semantic and to exploit this by the reasoning capacity inside the knowledge processing. Such a solution would need to describe each individual algorithm by features that are important characteristics that model its behavior. This information then could be treated to reason about the usefulness of a certain algorithm for a specific detection situation. The reasoning would have to be based on features of the objects in the scene and their importance for a processing decision. Thus, the semantics inside the processing network has to be connected by relations and rules to the scene knowledge.

Such an extended and more flexible connection between scene and processing domain needs extended experience with algorithms and their interaction with certain characteristics of objects and data. For this reason, an implementation has to be postponed until the experience needed is collected based on a realization of the built-in solution, explained at the beginning.

5

IMPLEMENTATION

In this chapter, we illustrate our approach through two use cases: object classification in the railway system and object detection inside an airport building. The goal in both cases was to detect and check relevant objects inside a defined work area. In the first example, a scene of the German Railway (DB), section from Nuremberg to Aschaffenburg, has been used. The railway equipments along the left and the right side of a rail track are scanned and presented as point clouds. Our approach was tested with the point clouds of 2500 meters long of the railway and knowledge about the scene provided by experts in railway system to classify the defined objects. The classification results are presented after that. In the second example, we used scanned data of the waiting area inside a building at Frankfurt International Airport in Germany (Fraport) for implementing a test for our method. The check-in area in Fraport is equipped with the objects of interest such as: separation walls, chairs, panels, recycling bins, etc., which were detected by using our approach. The obtained results are also shown in this chapter.

5.1/ OBJECT CLASSIFICATION IN THE RAILWAY SYSTEM

We dealt with scans in the vicinity of the tracks. Data were captured using a LIMEZ III, a special train equipped with a laser scanner mounted at its front-end. We selected a point cloud of 2500 meters railway which contains typical equipments of a railway system, for example: a station, electric pole, signals, advance signal marker posts, which are also objects of interest in our experiment. Besides, the used dataset includes “non-interest” objects (i.e. bridge, shrubs, heaps of stone, etc.) that challenge our algorithms in detecting and classifying target objects in the scene. The scanned data were split into five 500-meter-segments by the standard software tool of DB, we tested our system with each segment. The origin of the whole process can be seen as collecting, structuring and modeling all available knowledge. This includes the analysis of existing databases, guidelines, rules or other information available from the user side. More general knowledge (for example that a window is an opening inside a wall) has also been collected and modeled. This knowledge is entered into the knowledge base and expressed in interrelated ontology with rules, constraints and other components. Other relevant information (concerning probably existing individual objects) is also entered. Two non-domain experts worked for approximately 20 days to build the DB example ontology. During this period, the persons in charge of this task had to interact with two domain experts from the company. The work carried out during this period includes data extraction from various sources, the definition of objects and the scene knowledge, and the creation of relationships and rules.

The algorithm knowledge components are modeled and their possible connections are described in Fig. 5.8. The selection module checks the compatibility between inputs and outputs in order to create a graph (Fig. 5.8) of all possible algorithm sequences.

5.1.1/ KNOWLEDGE MODELING

Object detection and classification algorithms, which we model and use for particular scenes, vastly depend on the characteristics of a specific scene. In order to model scene knowledge, significant scene characteristics should be first collected. In the section, we introduce how the scene knowledge can be modeled as of different sources, from explicit knowledge like observation, rules, documents, CAD, GIS to implicit ones such as notions from experts in the rail system field. In the knowledge modeling process, we observe the usual railroad system – which is particularly applied in Germany – in both states: real scenario and scanned data (point clouds) to extract prominent features. Characteristics of the scanned data are also concerned. We then characterize the impact of scene knowledge and data knowledge on the detection algorithms through their intrinsic parameters. This process is based on the “trial-and-error” method.

Scene and data knowledge acquired from a railroad scene

- The railroad is a linear track with thousands meters in length, an algorithms is not capable to process entire scene in one time.

- The regions containing objects of interest (i.e. signals, electric pole, traffic lights, etc...) are two areas of 2 m width and located at the left and the right side of rail track (2 m width) and apart from the center of the track 1.57 m (see Fig. 5.1).

- The scanned data (point clouds) is devided into sections whose length should be long enough to conserve spatial relations between objects (i.e. two objects have a defined distance should not be in two different sections).

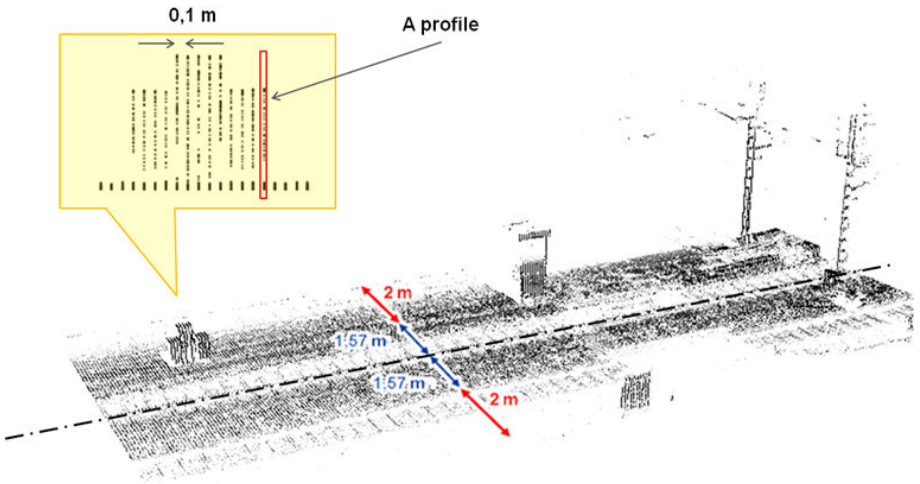


Figure 5.1: Particular characteristics in the railroad point cloud including regions of interest (2 m width) and distance between profiles (0,1 m)

- Point clouds are acquired from laser beams rotating 360 degrees (perpendicular to the rail track direction). The beams are projected from the scanners mounted on the head of a train moving along the rail track. The point clouds are therefore created from two

movements: rotation and translation form a shape similar to a circular polarization.

- At both sides of the track, objects are only scanned on one side: the side facing the rail way.

- One scanned circularity is a “profile”. On the visible face of an object in the point cloud, the distance between two continuous profiles is 0.1 m.

- Point clouds at ground do not affect the representation of objects in the vicinity of the track.

Geometrical features of objects

- Most objects of interest stand vertically to the ground and are located close to the track, for example a traffic signal in Fig. 5.2.

- Distance from objects to the track is relatively the same. Therefore point clouds density representing these objects has the same quality.

- Object has width less than 2 m and length often less than 1 m. Length is measured following the railroad direction while width is measured following the orthogonal direction to the tracks.

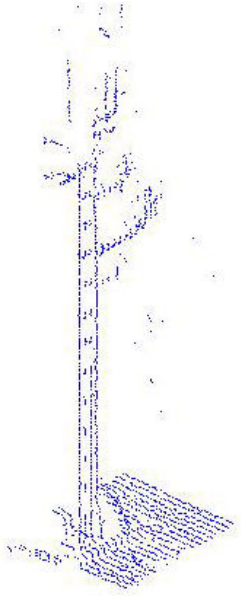


Figure 5.2: A traffic signal possesses the shape of a column constituted from linear structures

- Positions of some objects are often symmetrical, for examples: light signals are usually distributed as one on the left and the other on the right of the tracks.

- The visible surface of objects is presented by co-linear point sets along vertical direction.

- The geometric structure of an object often consists of many lines with different orientation and a few planes at the bottom. However, planes are not shown clearly in the scanned data.

- Due to the movement of scanner, the vertical lines in the object are captured and repre-

sented better than horizontal lines.

Parameter selection based on "Trial-and-error" method

Due to the variability of the scanned data quality and object characteristics (some geometries are described in the knowledge base), the detection results thus have different quality. This requires the numerical processing algorithms to adapt to different situations to gain the most accurate results, compare with ground truth. In each algorithm, we model one or more than one parameters which allow algorithms to adjust to yield different outcomes. The quality of results depends on how we alter parameters in the algorithm. We rely on the data knowledge and scene knowledge to alter parameters of an algorithm to gain proper outcomes. To do that, we employed "trial-and-error" to find the suitable values for each algorithm's parameters in particular situations.

Example 1: Parameters selection in the line detection algorithm

We present an example (Fig. 5.3) of selecting threshold values for a parameter in the line detection algorithm in 3D. The line detection algorithm is based on RANSAC whose employed threshold is the maximum distance from a point considered as a hypothetical inlier to the line model. From related documents in the DB domain as well as the knowledge from experts, the definition of objects is given. The definition includes geometry description of the object, such as number of linear structure in the object and geometry characteristics, for example, line characteristic: *thin, thick, continuous, disconnected, etc*. These create ground truth. Based on "trial-and-error", we implement several tests with different thresholds used.

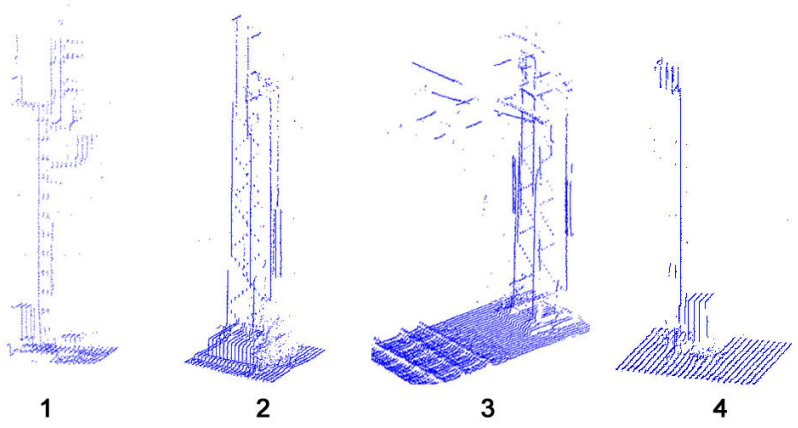


Figure 5.3: Four objects (1-4) in the railroad system

After a "trial-and-error" process, we employ the detected results in conjunction with the defined object geometry in the knowledge base to draw a suitable threshold. The decision of selecting a threshold value is usually made with the presence of the DB domain experts. In this example, we establish a correlation between thresholds and object, data characteristics as described in the table below:

Note that, one object may have different geometrical characteristics. For instance, both thin lines and thick lines may exist in an object. The reason is the diversity of object structures and/or quality of scanned data. One algorithm can therefore employ more than one threshold to detect an object. For example, to detect lines in the object #3 (contains both thin and thick lines), the line detection algorithm with threshold of 0.05 and 0.08 are

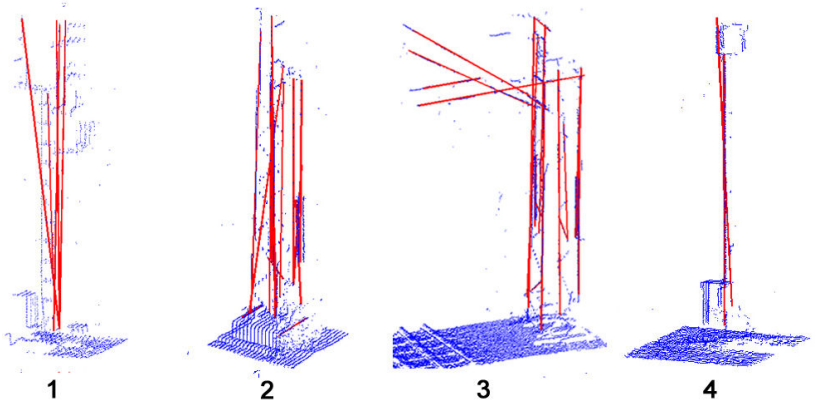


Figure 5.4: Results of line detection using RANSAC fitting algorithm with threshold 0.05

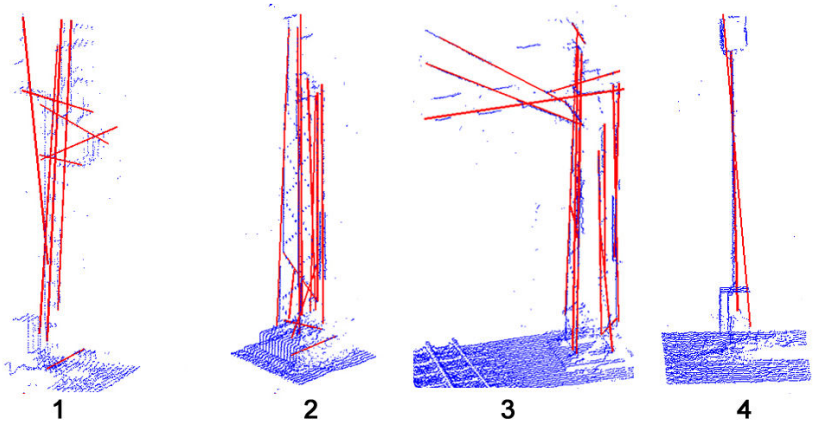


Figure 5.5: Results of line detection using RANSAC fitting algorithm with threshold 0.08

both executed.

Example 2: Parameters selection in the dimension approximation algorithm

The second example (see Fig. 5.6) is dimension approximation for a sub-point cloud. We use height algorithm to measure the height of an object represented in the point cloud. Basically, height approximation algorithm first seeks the mean point of a predefined number of highest points, then calculates the distance from the mean point to the ground.

In this example, the number of highest points in the detection of the objects #2 and #4 are selected as 10 since these objects have sharp details on the top. Number of highest points in case of detecting objects #1 and #3 is 20 as many outliers are included on the top of these objects.

Example 3: Parameters selection in the position detection algorithm

In the DB scene, some sections contain special objects such as a bridge, a roof (of a station) or a curbstone. The point clouds of such objects, when projected on the horizontal plane (ground plane), are usually represented as lines with various lengths. Our method detects separately positions of isolated objects (which are shown as points in the projection image) and such object with different lengths mentioned above. As the objects of interest are defined in the knowledge base, their lengths are therefore given. The position

Threshold	Object characteristic	Data characteristic
0.05	Thin line (object: #2, #3, #4) and simple structure: one single vertical element (object: #4)	High density, this shows the linear elements represented as continuous lines (object: #2, #3, #4) (Fig. 5.4).
0.08	Thick line. Particularly, in this data, many horizontal lines are represented as the thick lines (object: #1)	Point cloud has noise. The visibility of object is not clear or occluded (object: #1, #3) (Fig. 5.5).

Table 5.1: The correlation between object, data characteristic and threshold (in RANSAC line fitting algorithm)

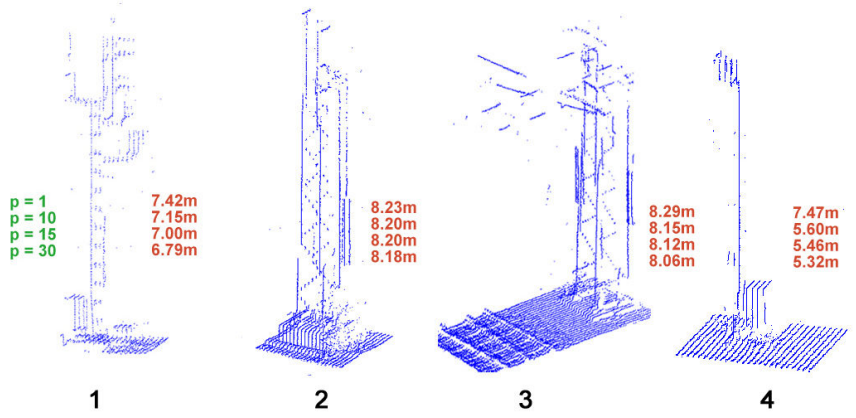


Figure 5.6: Various height values of objects obtained by differently selecting the number of highest point

detection algorithm is built with an extra function which allows to detect these lines in 2D (based on the line detection using Hough transformation).

The position detection algorithm has a parameter (takes Boolean values) which allows to manage when an line detection algorithm is needed. This extra algorithm detects points which lay on a line (i.e. blue lines in Fig. 5.7). These line segments are then classified based on the lengths of the defined objects such as bridges, stations or curbstones. The line with a specific length of the bridge (in this example) would be labeled as the position of the bridge.

5.1.2/ PROCESSING

The whole processing chain requires an initialization in order to detect entities and in the refinement steps. Such an initialization has to follow clear and prominent characteristics allowing to obtain reliable candidates for a first classification. The prominent characteristic is the most common one in the properties of the defined objects. In the DB case, a prominent property can be found in the vertical structure of most objects (for example: *electric poles* and *signals*). Such vertical structures are accessible via a vertical projection of the point cloud and an analysis of the resulting feature values. In a second step, feature values are analyzed in order to find evidence of sufficient characteristic entities.

Number of the highest points	Object characteristic	Data characteristic
$1 \leq p \leq 10$	Sharp on the top of object (object #2, #4)	Low noise
$10 \leq p \leq 20$	Complex structure on the top of object (object #1, #3). Data representation is ambiguous (#1, #2, #3)	Has noise and outliers

Table 5.2: The correlation between object, data characteristic and number of points selected in the volume approximation algorithm

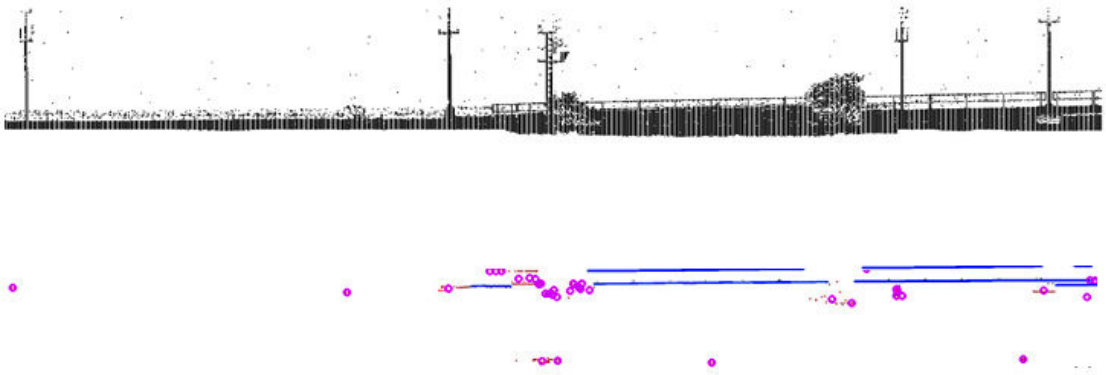


Figure 5.7: Projecting point clouds of a bridge on the ground plane, shape of the bridge is presented as lines in 2D

These entities are passed into the knowledge base along with their coordinates and other feature values to serve as an input for the classification step. Note that this step also allows identifying the ground in the scene. Although the ground may be considered as an object and hence could be passed into the knowledge base, only foreground objects were fed into the knowledge base in this case study. If the initial result did not allow for a classification, the algorithm's parameters are altered in this iteration. A refinement step attempts to detect additional characteristics of the entities found. The point cloud of an entity is therefore segmented into smaller "sub-point-clouds" which are checked for additional features. This step relies on the values of the most common feature to classify the detected entities. For instance, the knowledge base stores *hasHeight* as a common feature and an appropriate algorithm (*DimensionApproximation*) is selected to calculate the height of each entity. Based on the values of such feature, the available knowledge is used to classify the entities as:

Identified: as soon as a feature value is in the range of a class. This annotation has to be supported by subsequent classifications and remains valid as long as no conflict is detected.

Ambiguous: as soon as a feature value satisfies more than one class. Both annotations are stored and have to be separated by subsequent classifications and remain doubtful as long as no separation is possible.

Unknown: indicates that a feature value does not match any existing class. Further processing then requires the ASM to select other properties in order to continue the process.

Note that the label assigned to an entity may or may not change with every new iteration. Based on the state of this information, the ASM chooses the best suited algorithm for generating new characteristics, which will help in the next classification step. This selection also integrates the choice of an optimal sequence out of several possible ones (routes) of algorithms (or nodes). The aspect of quality can also be incorporated into the concept, for example; data: *noise, point density, point of view*; object: *size, shape, orientation*; scene: *possible objects, neighborhood, etc*. These factors may either be realized by thresholds modeling data noise or by changing the strategy of selecting a path through the graph. The latter case handles situations in which features are sensitive to noise and corresponding algorithms might fail. Although a simple example, assigning labels nevertheless shows the general logic, which can then be further extended with other considerations among entities. Success is directly related to the ability to detect entities and the significance of the feature values chosen. Less characteristic features can also be used. However, these will require more iterations and additional rules in order to achieve a stable classification.

Example: an electric pole (type 2) is represented by parallel vertical supports. ASM searches and selects the relevant algorithm - CheckParallel from the algorithmic library. This library is described by a graph (Fig. 5.8) representing all allowed connections, based on input and output between algorithms. Based on some data quality thresholds, the sequence may or may not include pre-processing algorithms (e.g. *NoiseReduction*). On the path from the starting algorithm (in this case, *PositionDetection*) to the desired algorithm (*CheckParallel*), ASM infers and invokes all concerned algorithms based on the *hasInput/hasOutput* property. *Segmentation, NoiseReduction and LineDetection1* are the selected ones. Afterwards, ASM links them together to create a proper sequence. It then looks as follows (result illustrated in Fig. 5.12, Fig. 5.13):

$$PositionDetection \rightarrow Segmentation \rightarrow NoiseReduction \rightarrow LineDetection1 \rightarrow CheckParallel$$

The execution of this sequence provides a list of recognized object entities which are then classified. Further sequences are used to improve the quality and to reduce the ambiguity within the results (Fig. 5.14). Iterations are repeated until a complete annotation for all entities is performed. The convergence conditions are applied to terminate the detection process for entities.

We have processed a 500 m section along the railway. Out of 12 algorithms modeled in the knowledge base (Fig. 5.8), the following ones were used by the system to classify objects (Tab. 5.3): *PositionDetection, Segmentation (cropping points surrounding a given position), DimensionApproximation, NoiseReduction, LineDetection1 (using RANSAC) and AngleCalculation*. Knowledge was collected carefully in order to provide a reliable knowledge base related to objects, scene, the nature of the data, algorithms and relationships between them.

Classification step: the ontology schema holds the semantics of the objects such as the nature of its geometries and 3D spatial characteristics. This information helps to identify the nature of detected entities using the inference capability of the knowledge tools. The complexity of the required rules directly depends upon the complexity of the situation to

Class	Object properties
Electric pole (type 1)	Vertical structure, height, perpendicular lines
Electric pole (type 2)	Vertical structure, height, parallel lines
Electric pole (type 3)	Vertical structure, height, oblique line
Main signal (Mechanical signal)	Vertical structure, height, perpendicular lines, parallel line, number of lines
Main signal (Light signal)	Vertical structure, height, perpendicular lines, parallel line, oblique line, number of lines

Table 5.3: Classes and properties used in DB scenario

be processed. In simple cases, even very simple rules are sufficient to produce a correct result. However, this concept also allows handling more complex situations. A simple classification of an entity (Geometry) based on a SWRL rule annotates an electric pole (type 2), as found along railway tracks:

$$\begin{aligned} Geometry(?x) \wedge hasHeight(?x, ?ht) \wedge swrlb : greaterThan(?ht, 4) \\ \wedge swrlb : lessThan(?ht, 6) \rightarrow ElectricPole2(?x) \end{aligned} \tag{5.1}$$

A first extension of such simple geometric considerations is possible by the use spatial relations. It only requires having the appropriate algorithms available and provides the result for the topological operation. In a simplified example, the following rule specifies that a “Building” defined in the ontology that overlaps a “Railway” (defined as well in the ontology), is a “RailwayStation”.

$$Building(?b) \wedge Railway(?r) \wedge topo : overlaps(?b, ?r) \rightarrow RailwayStation(?b) \tag{5.2}$$

Fig. 5.9 shows our process guided by various knowledge domains in object detection and classification. In this figure, object names are referred to as A, B, C. . . etc. We recall here that the process iterates until convergence (all objects are labeled) or stopping conditions (maximum number of iterations without refinement) are met.

5.1.2.1/ RESULTS

The base was progressively extended with new knowledge gained either from the analysis of the detected geometries or from classification results. Initially, 17 classes were defined as subclasses of the 5 classes in Tab. 5.3. These classes represent different types of signals and electric poles that can be found along the tracks and are of interest to our study. A total of approximately 500 geometries such as 3D line segments, angles and points of interest were recognized, 10 SWRL rules are used and 63 entities (possible object positions) were identified after the initialization step shown in (Fig. 5.11). All entities include possible objects in the scene but also noise and non-interest objects. The true number of railway objects was 13 (Tab. 5.4). With the second iteration, the process tries to refine the results and classify the objects. At the end, 10 out of 13 real railway objects were correctly classified, 50 entities which represent non-railway objects were classified as unknown, and 3 railway objects could not be unambiguously classified with the rules

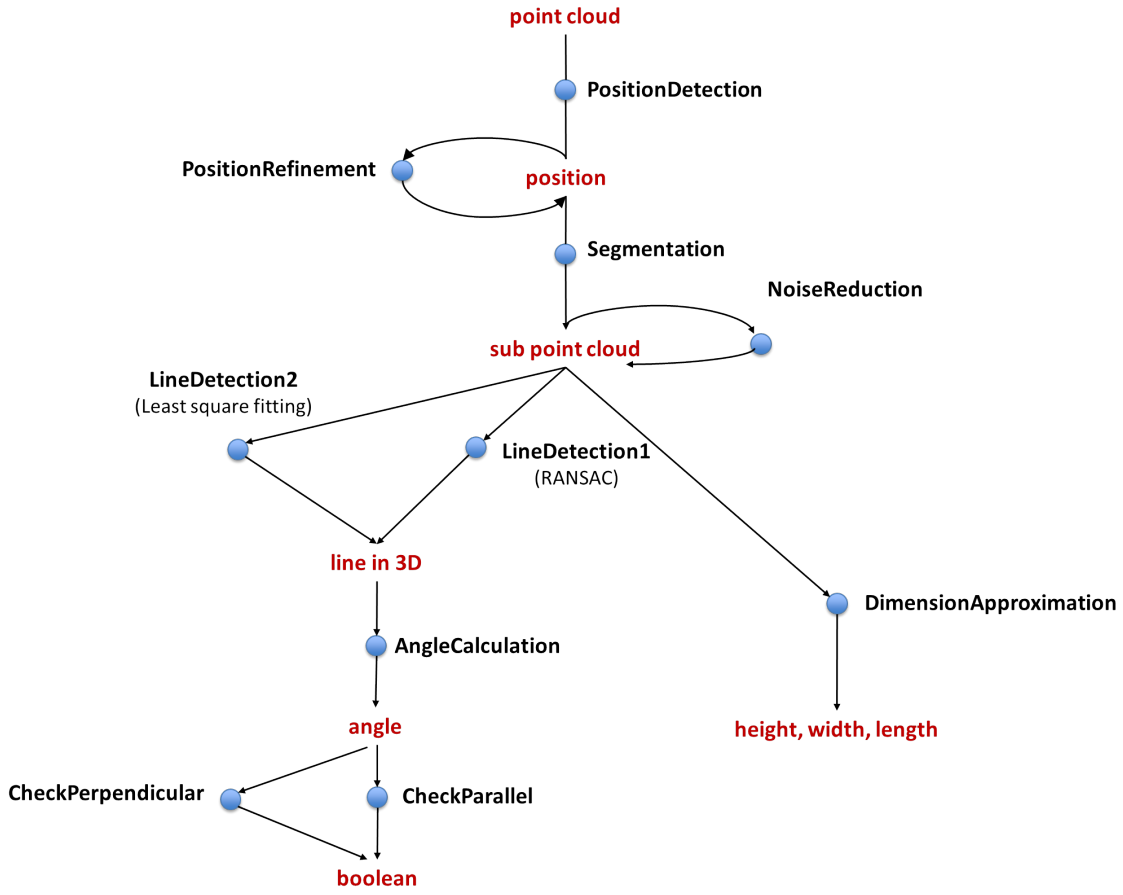


Figure 5.8: Graph of possible algorithmic paths generated by ASM and used for detecting objects in DB

implemented. The results in Fig. 5.14 were obtained by our software system. Computation took about 10 minutes on an Intel Xeon 2.4 GHz with 12G RAM. Note that our software is a prototype and has not been optimized for performance. In our experiments, we used the "shortest path" criterion from starting the algorithm to desired algorithm in order to find the optimal algorithm sequence. Our system assumes equal weights for all edges in the algorithms graph, i.e. factors that are intrinsic to algorithms such as time and memory requirements are not taken into account at this stage. Results can be improved by applying more complex rules, possibly using additional geometric constraints such as line or plane orientation, angle between lines or number of lines expressed in the rule 5.3:

$$\begin{aligned} &Geometry(?x) \wedge hasLine(?x,?l) \wedge line(?l) \wedge DistanceSignal(?y) \wedge DistanceFrom(?x,?y,?dis) \\ &\wedge swrlb : GreaterThan(?dis, 1000) \wedge hasHeight(?x,?h) \wedge swrlb : GreaterThan(?h, 4) \\ &\wedge hasVerticalLineNumber(?x,?vn) \wedge swrlb : lessThanOrEqual(?vn, 2) \\ &\wedge hasObliqueLineNumber(?x,?on) \wedge swrlb : equal(?on, 0) \rightarrow MainSignal(?x) \end{aligned} \tag{5.3}$$

In order to relate the classification to human interpretation the point cloud was presented to test persons. They identified 8 of 13 railway objects based on a visual inspection of

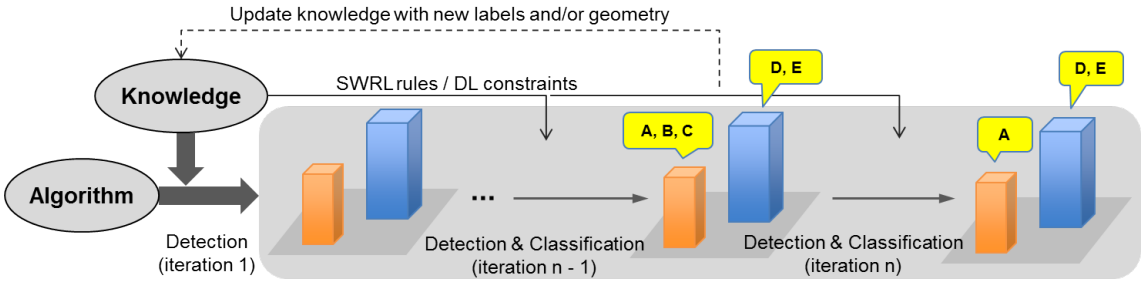


Figure 5.9: Knowledge-driven method for object detection and classification process

Object	Visual inspection	Knowledge-based data processing
Electric pole (type 1)	1/1*	1/1
Electric pole (type 2)	2/4	4/4
Electric pole (type 3)	1/1	1/1
Main signal (Mechanical signal)	2/4	2/4
Main signal (Light signal)	2/3	2/3
Total	8/13 (61,53%)	10/13 (76,92%)

Table 5.4: Experiment in a section of DB railway, comparison result between two approaches: Visual inspection using the standard software tool of DB, and knowledge-based data processing

the cloud and without taking into account topological or descriptive knowledge. This just shows the limited representation of objects inside such types of point clouds. One major reason for the poor quality of the point cloud is the fact that only the side of the object facing the tracks is captured due to the scanner on the train. However, this also shows the usefulness of additional knowledge.

(*) Number of detected objects over number of ground-truth objects.

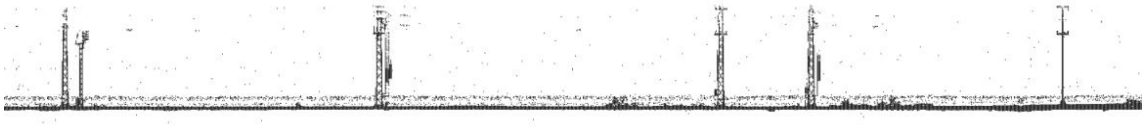


Figure 5.10: A railroad segment representing the objects of interest such as: light signals, distance signals, electrical poles, and advance signal marker posts

Results obtained after the processing along the tracks are shown in Tab. 5.4. Note that the only failures using knowledge were *Main signals* that could also not be recognized by visual inspection. This is mainly caused by the poor quality of the data, especially in terms of point density, which made such structures hardly visible and undistinguishable. The type 2 electric pole was successfully identified using the automated detection and classification whereas visual inspection failed.

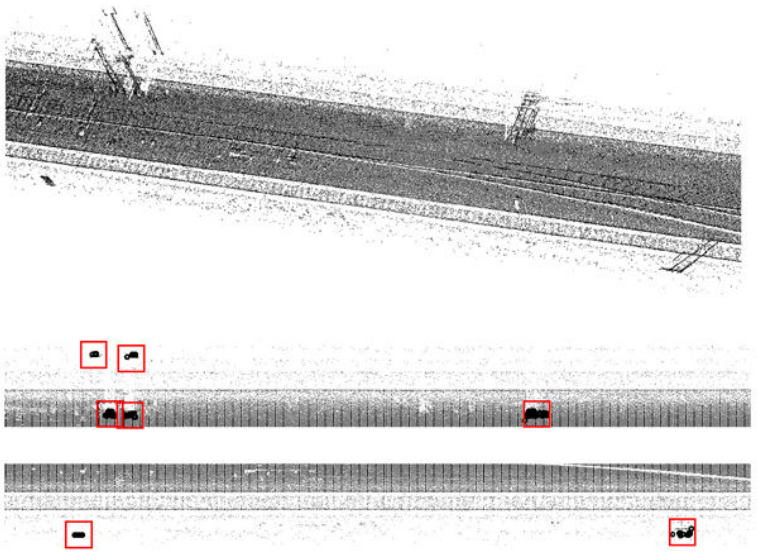


Figure 5.11: Point cloud representation of a section of a railway (top). Results after executing the initialization step, projecting the point cloud to the ground plane, rectangles denote possible object positions (bottom)

5.2/ Object detection inside airport building (Fraport's waiting area)

In the second case, we used scans from an environment inside the airport buildings, typically a waiting area. Changes in the technical infrastructure were of main interest. Data were obtained from classical terrestrial laser scanning. The Fraport scenario is an indoor architecture of a waiting room in a boarding area of Frankfurt airport. It contains regular walls, floor, chairs, advertisement panels, signs etc. The whole scene has been scanned using five terrestrial laser scanners and registered, resulting in a large point cloud representing the surfaces of the scene objects captured from different scanning positions (Fig. 5.15). This scenario is different from the DB test example because a data base of expected objects in the scene exists and can be used as a prior knowledge. Two persons worked for about 10 days to fill the ontology with knowledge such as properties of objects, scene, nature of data and characteristics of buildings. The data sources were CAD plans, related documents from the experts and observations from the real scene.

5.2.1/ Processing

The main issue in the Fraport case is to detect objects inside a building which are probably moved and changed their positions due to different purposes of users. However, we admit that some static objects such as walls, separation or advertising panels, etc... cannot be moved. Therefore, our strategy was first attempted to validate the presence of static objects in the point cloud that were supposed to exist according to the data base. After that, moveable objects like chairs, trash bins, were detected and also fed into the knowledge base. The initialization was different from the DB case because of more

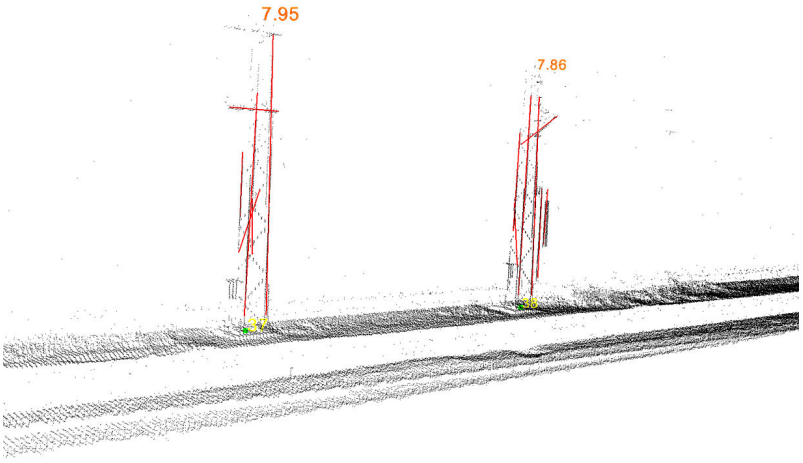


Figure 5.12: Results from detecting 3D lines of a signal and electric pole (type 3) along the railway

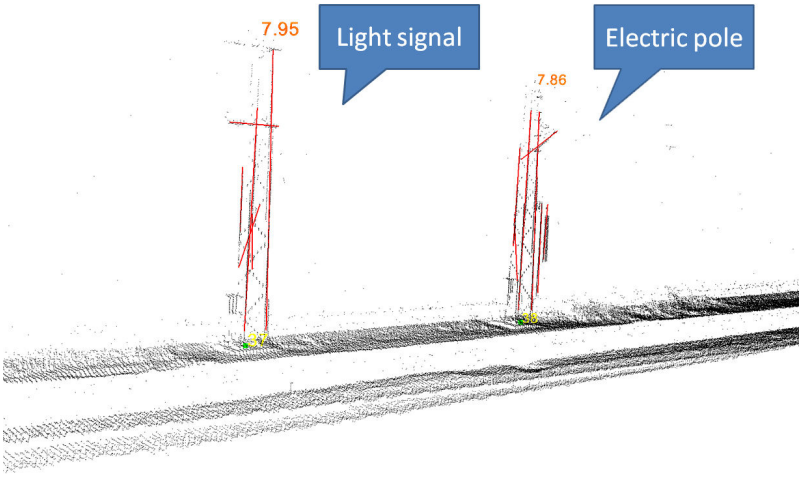


Figure 5.13: A classification and identification process based on the detected results. Two objects were identified correctly as “Light signal” and “Electric pole”.

complex objects and the prominent role of many vertical planes.

Static objects detection

Walls are mostly fixed in the waiting area of Fraport. The presence of walls and such vertical elements in the point cloud had to first be validated. By projecting the point cloud onto the ground plane, we obtain a projection image which represents the footprint of objects in the scene. Vertical plane detection was possible by a vertical projection of the point cloud followed by *Hough Line detection* to locate the static objects' position on the ground plane. *VerticalProjection* and H*oughLineDetection* are included in *PositionDetection* algorithm. Points with a vertical projection in the vicinity of these lines were used to define segments corresponding to vertical planes. We use a bounding box to represent the point cloud of a vertical plane. Bounding box covers entirely point clouds of the vertical plane and is determined by eight vertices (Fig. 5.16). The following step was used to verify walls, separation panels or advertising panels defined in the data base based on their particular length, height and width (Fig. 5.17).

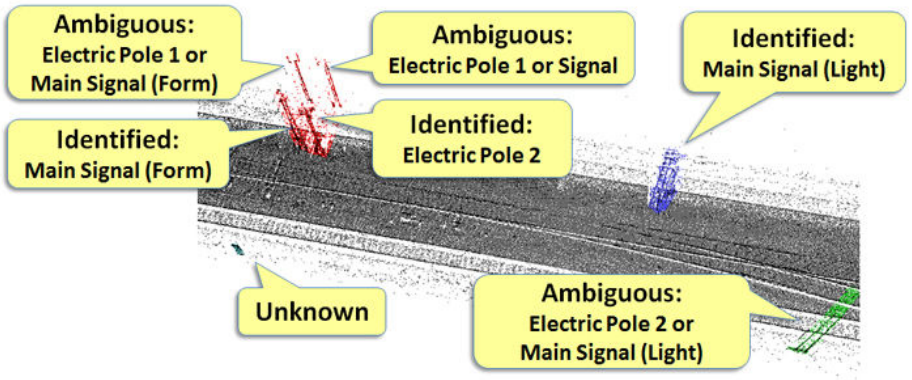


Figure 5.14: Positions of objects and annotation results after the first iteration

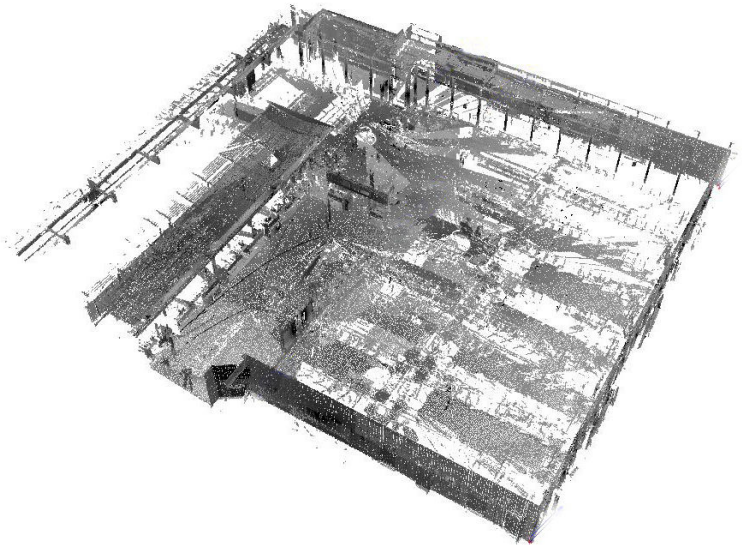


Figure 5.15: 3D scan of a check-in area inside the Fraport

For example, to classify walls among the detected vertical planes, we based on the knowledge in the Fraport case to draw a rule: *"Select all vertical planes (geometries) whose position were determined, the geometry which has height greater than 3 m, width greater than 0.1 m and length greater than 4 m is a wall"*. This rule can be expressed in a form of SWRL as:

$$\begin{aligned} &Geometry(?x) \wedge hasCorrespondingGeo(?x, ?v) \wedge VerticalPlane(?v) \wedge hasHeight(?x, ?hei) \\ &\wedge swrlb : greaterThan(?hei, 3) \wedge hasWidth(?x, ?wid) \wedge swrlb : greaterThan(?wid, 0.1) \\ &\wedge hasLength(?x, ?len) \wedge swrlb : greaterThan(?len, 4) \rightarrow Wall(?x) \end{aligned} \tag{5.4}$$

All vertical planes satisfied the rule above would be stored as "walls" in the ontology.

Moveable object detection

Class	Object properties
Wall	Vertical plane, length, height
Separation panel	Vertical plane, length, height
Advertising panel	Vertical plane, length, height, number of planes
Chair	Horizontal plane, leaning plane, angle between planes, length of chair

Table 5.5: Classes and properties used in the Fraport scenario

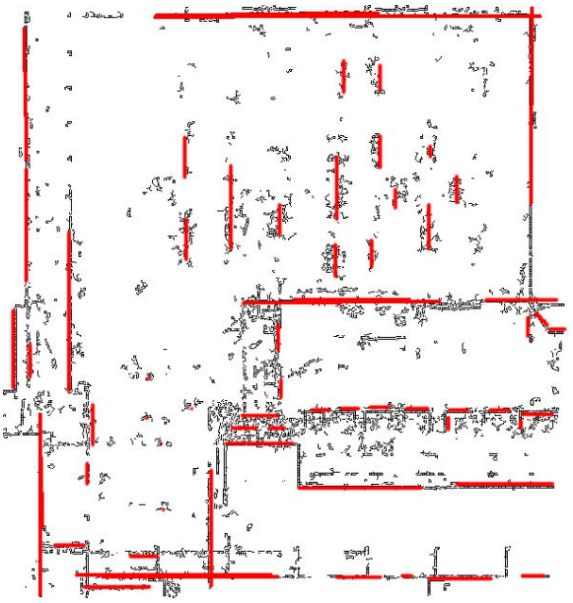


Figure 5.16: Detected line segments corresponding to vertical planes in 3D

There are also many moveable objects like chairs, tables, counters, or trash bins, which also need to be detected to update the knowledge base. All objects already available from the first validation phase gave a geometric and semantic frame helping to support the detection of unknown moveable objects. We focused on detecting walls in the border region of the check-in area. Only two walls exist in the scene and the remaining larger static structures are either separation or advertising panels, which are easily distinguishable from walls by their specific height. Both walls were successfully identified. Walls gave a semantic frame to support the detection of the moveable objects. For example, chairs were searched for in a specific area defined within a certain distance from the wall and a certain height above the floor. Note that the reference frame of our point cloud is attached to the floor such that the latter is simply determined by fitting a horizontal plane (initialized at height Z = 0) using the PlaneDetection algorithm.

In this example, chair sets were found in a specific area (C) (Fig. 5.19), and defined within a distance of 5 m from the walls and 0.7 m above the floor. Because chair sets were arranged in a predictable parallel pattern, they were detectable through a division of the point cloud (C) into sub-point clouds.

Chair identification

The chair definition (stored within the knowledge base) consists of features, geometries,

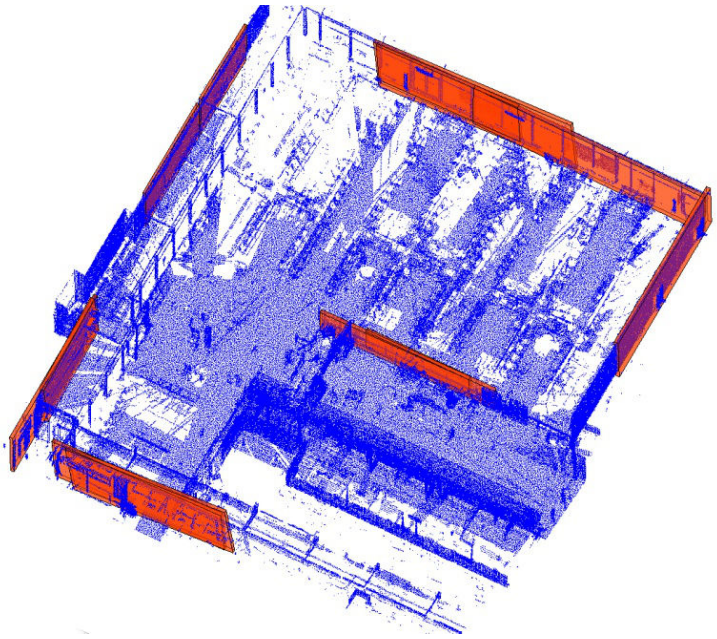


Figure 5.17: Walls are detected based on the rule

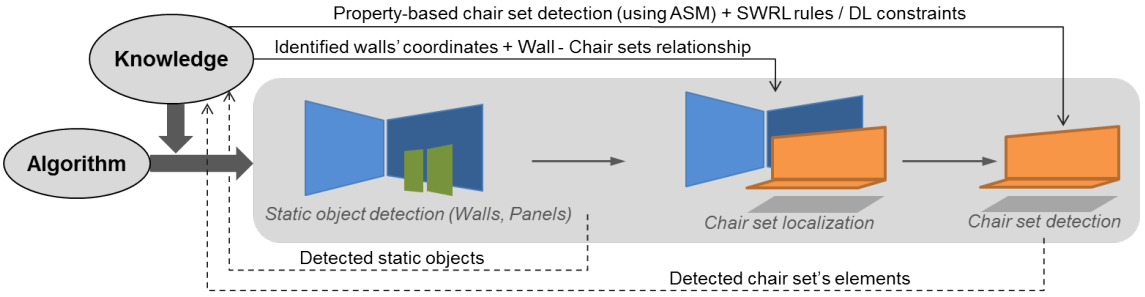


Figure 5.18: Chair set detection process

and compositions, such as chair lengths, seat planes, leaning planes and the angle between them. The chair as shown in Fig. 5.20b contains a composition of seat (normal vector $\vec{n_s}$) and a leaning plane (normal vector $\vec{n_l}$), with an enclosing angle of 120 degrees. The seating and leaning plane heights h_s and h_l are defined in the knowledge base. Based on this knowledge, the Plane Detection algorithm in conjunction with two defined normal vectors was used to detect the chair planes. Note that the leaning and seat plane point clouds have different levels of quality (e.g. different point densities), and that the geometries have particular characteristics (*thin, thick, etc*). Thus, to detect such planes, the PlaneDetection algorithm should perform accordingly. Another role of the ASM is to derive suitable values to PlaneDetection's parameters in the specific situations. Consequently, the ASM generated, based on the properties of a chair, an appropriate sequence of algorithms to invoke:

$$PositionDetection \rightarrow Segmentation \rightarrow PlaneDetection \rightarrow$$
$$DimensionApproximation \rightarrow AngleCalculation \rightarrow FitChair$$

The detected geometries are populated into the ontology, and the rules 5.5 in the knowl-

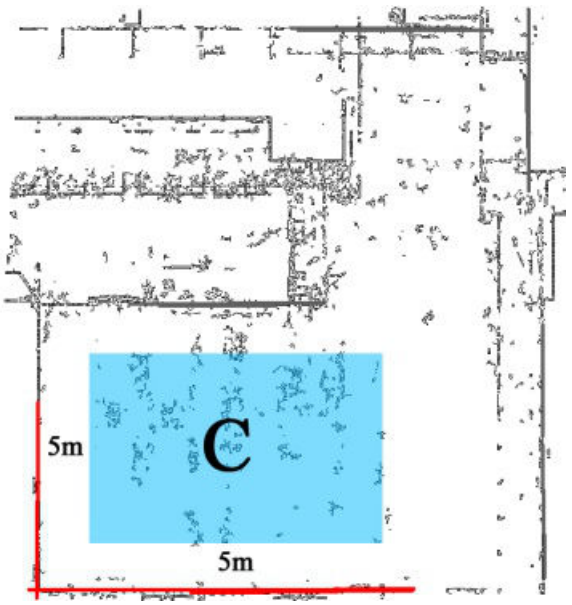


Figure 5.19: The chair area is found based on two detected walls.

edge base are applied to classify a chair:

$$Geometry(?x) \wedge hasCorrespondingGeo(?x, ?l) \wedge LeaningPlane(?l) \wedge hasCorrespondingGeo(?x, ?s) \wedge HorizontalPlane(?s) \wedge hasAngle(?x, 120) \wedge hasLength(?x, ?len) \wedge swrlb : greaterThan(?len, 370) \wedge swrlb : lessThan(?len, 380) \rightarrow Chair(?x) \quad (5.5)$$

Chair sets are arranged parallel to the walls and represented by very sparse point clouds (Fig. 5.20a). Nevertheless, it is possible to detect, model and identify chair sets based on a sequence of algorithms making use of topological and geometrical constraints arising from previously detected elements. Six algorithms were used (out of the 10 in Fig. 5.21) such as: *PositionDetection, Segmentation, DimensionApproximation, PlaneDetection (based on RANSAC), AngleCalculation and FitChair* (which verifies a chair by two connected planes in an angle of 120 degrees).

5.2.1.1/ RESULTS

The results obtained are shown in Fig. 5.12 and Fig. 5.23 in which the five chair sets 8-12 were successfully identified, the five chair sets 3–7 were only partly detected and the two chair sets ”1” and ”2” could not be identified due to missing points. In the next stage of processing, objects were verified using topological constraints, such as a distance-based identification from the identified objects. Finally, 10 out of 12 chair sets could be correctly classified even in an insufficient dataset. The results reported here were obtained with an ontology that had been filled with approximately 350 detected geometries (planes, line segments...) and used 4 SWRL rules. The process took about 7 minutes on an Intel Xeon 2.4GHz with 12G RAM when using our prototype software. The full process of detecting chair sets including wall identification is depicted in Fig. 5.18.

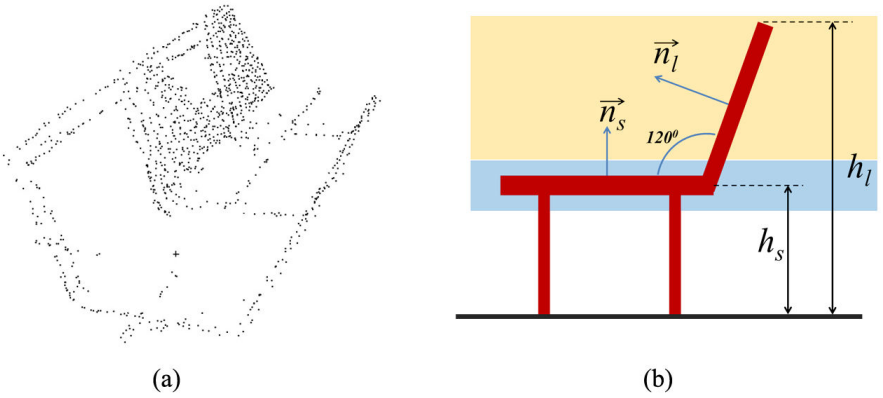


Figure 5.20: (a) Point cloud of a chair and (b) chair as a composition of seat and leaning planes

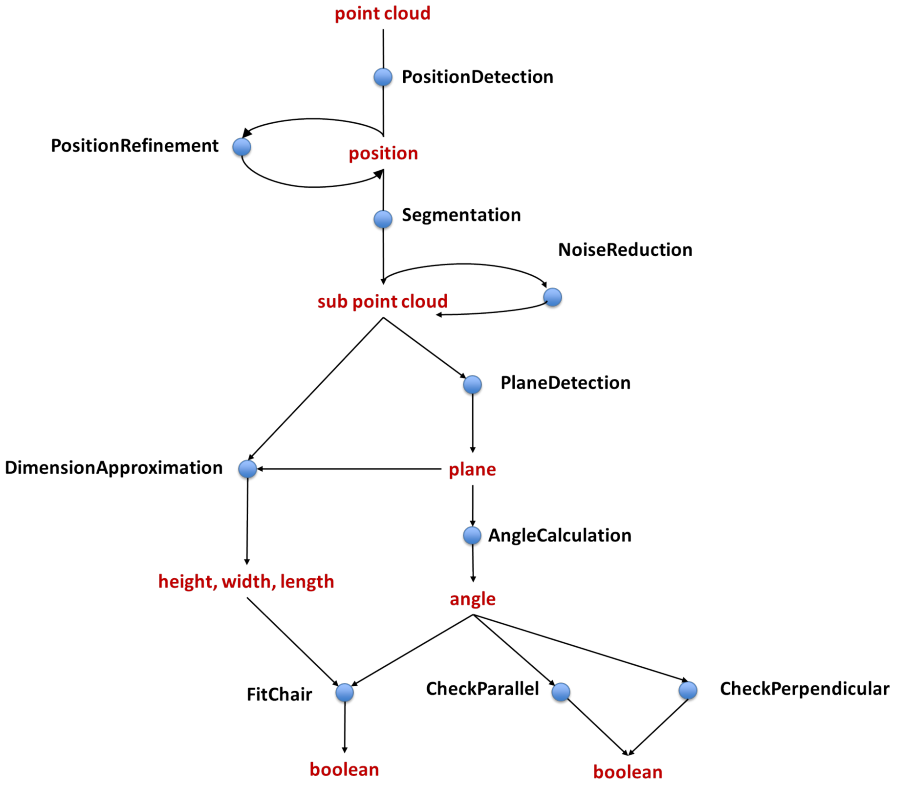


Figure 5.21: Graph of possible algorithmic paths generated by ASM and used for detecting objects in Fraport

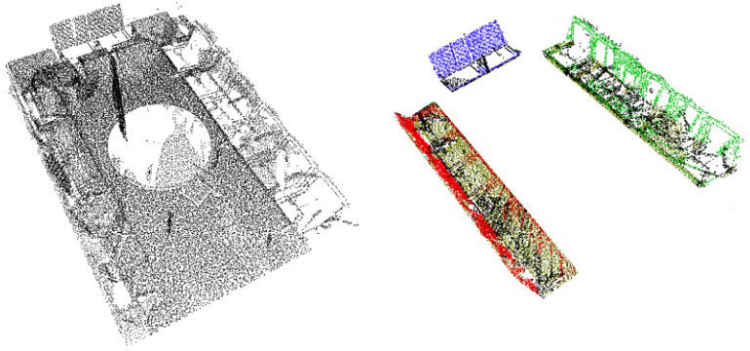


Figure 5.22: (a) A chair set point cloud and (b) a detected chair set

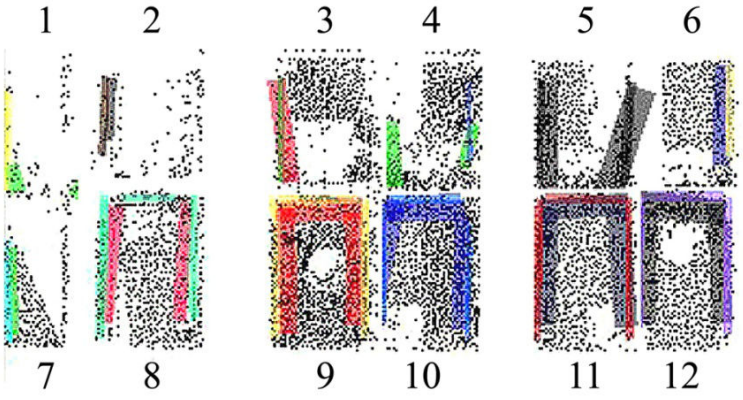


Figure 5.23: Identification results obtained on 12 chair sets in a waiting area (failures 1-2, partial detection 3-7, successful identification 8-12)

6

Conclusions and Future Work

The thesis presented a knowledge-driven approach to detect objects in point clouds. The approach was based on semantics of different associated domains which assist in the detection and classification of objects. Knowledge supported all processing steps, including the guidance of the data processing. This allowed inter-relating the characteristics of algorithms with those of the objects in the domain of the application. Our system also provided the flexibility to infer the strategy from existing knowledge and to adapt the processing to the application-specific requirements. In particular, the ASM, which uses semantics between algorithms and other domains to suggest appropriate algorithms or algorithm sequences, was an essential component within this platform. The first step was to detect object geometry, which needs to incorporate various factors from different knowledge domains. Indeed, the object geometries and their influencing factors or the dataset nature could alter algorithms or their parameters. The knowledge base allowed semantic investigation and classification suggestion through the ASM.

The permanent interaction between the algorithms and the knowledge base allowed for a smooth and gradual construction of the knowledge base. Such base contains at the end of the process all entities which could be detected and identified. Although knowledge needs to be provided at the beginning of the process, it only has to be collected once and then becomes permanently available for a certain application. In addition, the knowledge base can be iteratively extended by the operator observing the behavior of the system within various practical situations.

6.1/ Results

To summarize the work we have carried out in this thesis, some achievements we have reached are highlighted as following:

- A summary of the previous studies helps to evaluate appropriate methods to be used in our approach. Accordingly, our approach, under the consideration of particular environments, employed cutting edge methods to efficiently solve given issues. We did not only make use of existing algorithms, but we have integrated and combined robust algorithms to serve our purpose more effective (i.e. RANSAC and least-squares fitting in conjunction with data partitioning in extracting primitive shapes in an unordered data).

- A library of numerical processing algorithm was established to carry out different tasks in data processing as well as object classification. Each single algorithm was modeled to not only process in point cloud data sets but also comprehend the knowledge extracted

from the particular characteristics of data as well as objects. The algorithms are linked and represented as a directed graph. We created the ASM to allow individual algorithms to be connected in a proper way to for an algorithm sequence. While a single algorithm could perform a specific task, a generated sequence was able to carry out multiple tasks. The combination of algorithms created flexibility and intelligence in the system.

- Knowledge extracted from various domains contributed to the performance of an individual algorithm but also to all steps of a detection process. In a single algorithm, knowledge drawn from the semantics of data and object characteristics first selected a relevant algorithm among the available ones in the library. Second, knowledge controlled algorithms to adapt with different data conditions. In particular, this has been done by selecting appropriate parameter values for the algorithm. At each step of a detection process, the knowledge base stores the results after detection. These results were not only statically stored but also updated and frequently used to classify objects (based on predefined rules). Therefore, our system is an automatic object detection platform which is able to dynamically exchange information between knowledge and data processing in any or whole process.

- Our method was implemented through an industrial project (WiDOP). The project aimed at detecting objects in point clouds and classifying them based on the guidance of knowledge. The scenes and their data are captured in outdoor environment - the German Railway system - and an indoor one - Frankfurt International Airport. The two scenes have differences in object distribution on the ground (because of particular features of the test scene) and in data characteristics that are caused by both sensor characteristic and measurement method. The solution consisted in extracting and modeling knowledge domains, such as scene knowledge, data knowledge, spatial knowledge and algorithm knowledge to describe the semantic behavior of the processing algorithms. Semantic knowledge played an important role in selecting and guiding the processing algorithms to perform differently behaviors in each particular situation. Through two case studies, our method shown its ability to recognize objects in challenging scanned data guided by knowledge.

The results reported in the two use cases showed applicable of our approach in two different environments; indoor and outdoor scene. The quality of the results depends on the robustness of the implemented algorithms, the selected strategy and the amount of knowledge integrated. In practice, the solution is oriented towards the requirements of a specific application. Through the outcomes, our solution showed strength as well as weakness of a knowledge-based object detection system.

The representation of object in point clouds seems to be an important point of concern which tremendously affects the algorithms. The presence of knowledge in guiding algorithms following an intelligent manner and the robustness of the numerical processing algorithms have significantly enhanced the system's performance. The strength of the system is manifested by the fact that it was able to automatically detect and classify objects in different data conditions. The results have shown that our system is able to recognize objects in challenging conditions such as low density point clouds, occlusion (objects in the DB example), visibility low, even a part of object being not represented (chairs in the Fraport example). The results were also improved (compared to the conventional approaches') in the sense that knowledge supported for classification process is likely to increase the reliability.

In contrast, our approach was not able to gain accurate results in some cases. This was

for example the case with line segments extracted from point clouds, particularly data containing noise and or having low density. This may also have been caused due to an inappropriate setting of parameter values to the used algorithms. The “trial-and-error” process, which we have used for some data sets, is not sufficient to obtain full range of thresholds for all algorithms. Our system also failed in case of no reliable geometry detected in an object, consequence was that object have not annotated. The object definition in the knowledge, which is mostly done by knowledge engineers or experts, was not always correct. This led to classification failures even when the detection results were accurate.

6.2/ FUTURE WORK

Our proposed approach does not use pure-numeric strategies with fault-tolerant methods. Instead, it relies on human knowledge and experience for object detection and identification. We have implemented our approach in a prototype version which however should be improved to get rid of existing shortcomings. To do that, further work is needed:

- Further development is desired to make algorithms more robust to quality variations in the data, and to segment more complex objects. Furthermore, the “trial-and-error” process is also required to do many examples, including diversity of data quality as well as object type.

- The knowledge sources (data features, object properties and scene characteristics) have to be extended in order to enhance the classification processing, especially regarding ambiguous cases.

- Both an expansion of the ontology and further implementation and testing of rules are currently considered and subject to investigation. This will also require further tests using various datasets in order to achieve stable results and parameter values.

LIST OF MY PUBLICATIONS

Frank Boochs, Andreas Marbs, Helmi Ben Hmida, Hung Truong, Ashish Karmacharya, Christophe Cruz, Adlane Habed, Christophe Nicolle, and Yvon Voisin. Integration of knowledge to support automatic object reconstruction from images and 3d data. *International Multi-Conference on Systems, Signals and Devices (SSD)*, pages 1–13, 2011.

Andreas Marbs, Frank Boochs, Helmi B. Hmida, and Hung Q. Truong. Wissensbasierte objekterkennung in 3d - punktwolken und bildern. *Conference on DGPF-Tagungsband, 3-Ländertagung D-A-CH*, pages 220–227, 2010.

Hung Truong, Frank Boochs, Adlane Habed, and Yvon Voisin. A knowledge-based approach to the automatic algorithm selection for 3d scene annotation. *11th International Conference on Information Science, Signal Processing and their Applications (ISSPA)*, pages 225–230, 2012.

Hung Truong, Helmi Ben Hmida, Frank Boochs, Adlane Habed, Christophe Cruz, Yvon Voisin, and Christophe Nicolle. Automatic detection and classification of objects in point clouds using multi-stage semantics. *Journal of photogrammetry, remote sensing and geoinformation processing (PFG)*, 2013a.

Hung Truong, Ashish Karmacharya, Waldemar Mordwinzew, Celeste Chudyk, Frank Boochs, Adlane Habed, and Yvon Voisin. Automatic object detection in point clouds based on knowledge guided algorithms. *The international society for optics and photonics (SPIE)*, 2013b.

Hung Q. Truong, Helmi B. Hmida, Andreas Marbs, and Frank Boochs. Integration of knowledge into the detection of objects in point clouds. *International Society for Photogrammetry and Remote Sensing (ISPRS)*, XXXVIII, 2010.

Bibliography

Maryam Alavi and Dorothy E. Leidner. Review: Knowledge management and knowledge management systems: Conceptual foundations and research issues. *MIS quarterly*, pages 107–136, 2001.

A. Alharthy and J. Bethel. Detailed building reconstruction from airborne laser data using a moving surface method. *Arch. Photogrammetry and Remote Sensing*, XXXV, 2004.

B. Ameri and D. Fritsch. Automatic 3d building reconstruction using plane-roof structures. *American Society of Photogrammetry and Remote Sensing*, 2000.

D. Anguelov, B. Taskar, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A. Ng. *Discriminative learning of Markov random fields for segmentation of 3D scan data*. 2005.

Sean Bechhofer, Frank Van Harmelen, Jim Hendler, Ian Horrocks, Deborah L McGuinness, Peter F Patel-Schneider, Lynn Andrea Stein, et al. Owl web ontology language reference. *W3C recommendation*, 10:2006–01, 2004.

Aurilla Aurelie Arntzen Bechina and Martin Nkosi Ndlela. Success factors in implementing knowledge based systems. *Electronic Journal of Knowledge Management 7. 2*, 2:211–218, 2009.

Bit-tech. Artificial intelligence, perception, and achievements, 2012. URL http://www.bit-tech.net/bits/2012/03/19/aisearch/3.

L Bornaz and F Rinaudo. Terrestrial laser scanner data processing. In *XXth ISPRS Congress Istanbul*. Citeseer, 2004.

M. Bredif, D. Boldo, M. Pierrot-Deseilligny, and Maitre. 3d building reconstruction with parametric roof superstructures. *IEEE International Conference in Image Processing*, 2007.

H. Cantzler, R. Fisher, and M. Devy. Quality enhancement of reconstructed 3d models using coplanarity and constraints. *Pattern Recognition*, pages 34–41, 2002.

RC Chakraborty. *Artificial Intelligence*. Jaypee University Lecture, 2010.

Simon Colton. *Knowledge Representation*. Imperial College London Lecture, 2010.

Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):603–619, 2002.

Thomas H. ; Prusak Davenport and Laurence. Working knowledge: How organizations manage what they know. *Harvard Business Press*, 2000.

Konstantinos G Derpanis. Overview of the ransac algorithm. Technical report, Technical report, Computer Science, York University, 2010.

Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

Y. Duan, C. Cruz, and C. Nicolle. Managing semantics knowledge for 3d architectural reconstruction of building objects. *International Conference on Software Engineering Research, Management and Applications (SERA)*, pages 121–128, 2010.

N. Durand, S. Derivaux, G. Forestier, C. Wemmert, P. Gancarski, O. Boussaid, and Puissant A. Ontology-based object recognition for remote sensing image interpretation. *Tools with Artificial Intelligence (ICTAI)*, pages 472–479, 2007.

Staffan Ekvall, Danica Kragic, and Frank Hoffmann. Object recognition and pose estimation using color cooccurrence histograms and geometric modeling. *Image and Vision Computing*, 23(11), 2005.

Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

Doctor George. The math forum, drexel university, 2005. URL http://mathforum.org.

N. Goerke and S. Braun. Building semantic annotated maps by mobile robots. *Towards Autonomous Robotic Systems*, pages 149–156, 2009.

A. Golovinskiy, V. G. Kim, and T. Funkhouser. Shape-based recognition of 3d point clouds in urban environments. *IEEE 12th International Conference on Computer Vision*, pages 2154–2161, 2009.

P. Gottschalk. Toward a model of growth stages for knowledge management technology in law firms. *Informing Science*, pages 79–93, 2002.

Ronald L. Graham. An efficient algorith for determining the convex hull of a finite planar set. *Information processing letters*, 1(4):132–133, 1972.

Stephan Grimm, Pascal Hitzler, and Andreas Abecker. Knowledge representation and ontologies – logic, ontologies and semantic web languages, 2007.

S. Growe and R. Tonjes. A knowledge based approach to automatic image registration. *International Conference on Image Processing*, 3:228–231, 1997.

Thomas R Gruber et al. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220, 1993.

Donald D Hearn, M Pauline Baker, and Warren Carithers. *Computer graphics with open gl*. Prentice Hall Press, 2010.

V. Hedau, D. Hoiem, and D. Forsyth. *Recovering the spatial layout of cluttered rooms*. 2009.

V. Hedau, D. Hoiem, and D. Forsyth. Thinking inside the box: Using appearance models and context based on room geometry. pages 224–237, 2010.

H Hoefler, C Baulig, A Blug, H Woelfelschneider, O Fleischhauer, H Wirth, J Meier, C Lehmkuehler, and H Lenz. High speed clearance profiling with integrated sensors. In *World Congress on Railway Research (WCRR)*, 2006.

Ian Horrocks, Peter F Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, Mike Dean, et al. Swrl: A semantic web rule language combining owl and ruleml. *W3C Member submission*, 21:79, 2004.

B. Kahn and E. Adams. Sales forecasting as a knowledge management process. *The Journal of Business Forecasting*, pages 19–22, 2000.

Louay Karadsheh, Ebrahim Mansour, Samer Alhawari, Ghassan Azar, and Naser El-Bathy. A theoretical framework for knowledge management process: towards improving knowledge performance. *Journal of Communications of the IBIMA 7*, 2009.

Holger Knublauch, Ray W Fergerson, Natalya F Noy, and Mark A Musen. The protege owl plugin: An open development environment for semantic web applications. In *The Semantic Web–ISWC 2004*, pages 229–243. Springer, 2004.

Danica Kragic and Henrik I. Christensen. Model based techniques for robotic servoing and grasping. *In Intelligent Robots and Systems*, 1:299–304, 2002.

F. Lafarge, X. Descombes, J. Zerubia, and M. Pierrot-Deseilligny. Automatic building extraction from dems using an object approach and application to the 3d city modelling. *(ISPRS) Journal of Photogrammetry and Remote Sensing*, 63:365–381, 2008.

D. Lee, A. Gupta, M. Hebert, and T. Kanade. Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. *Advances in Neural Information Processing Systems*, 24, 2010.

N. Maillot and M. Thonnat. Ontology based complex object recognition. *Image and Vision Computing. 26*, 26(1):102–113, 2008.

Takashi Matsuyama. Knowledge-based aerial image understanding systems and expert systems for image processing. *IEEE Transactions on Geoscience and Remote Sensing*, 3:305–316, 1987.

Leonard McMillan. *Projection Transformations*. MIT Computer Science and Artificial Intelligence Laboratory (CSAIL), MIT, 2005.

O. M. Mozos. *Semantic Place Labeling with Mobile Robots*. 2008.

Chris Nikolopoulos. *Expert systems: introduction to first and second generation and hybrid knowledge based systems*. Marcel Dekker, Inc., 1997.

A. Nuechter, O. Wulf, K. Lingemann, J. Hertzberg, B. Wagner, and H. Surmann. 3d mapping with semantic knowledge. *RoboCup*, pages 335–346, 2006.

Kei Okada, Mitsuharu Kojima, Satoru Tokutsu, Toshiaki Maki, Yuto Mori, and Masayuki Inaba. Multi-cue 3d object recognition in knowledge-based vision-guided humanoid robot system. *International Conference on Intelligent Robots and Systems*, pages 3217–3222, 2007.

Robert M. O'keefe and Alun D. Preece. The development, validation and implementation of knowledge-based systems. *European Journal of Operational Research*, 92(3):458–473, 1996.

OTEC. Data, information, knowledge, and wisdom, 2007. URL http://otec.uoregon.edu/data-wisdom.htm.

Dan Patterson. *Introduction to artificial intelligence and expert systems*. Prentice-Hall, Inc., 1990.

M. Pollefeys, R. Koch, M. Vergauwen, and Van Gool. Automated reconstruction of 3d scenes from sequences of images. *ISPRS Journal Of Photogrammetry And Remote Sensing*, 55(4):251–267, 2000.

I. Posner, D. Schroeter, and P. Newman. Using scene similarity for place labelling. *In 10th International Symposium on Experimental Robotics*, 39:85–98, 2008.

Protege. The protege ontology editor and knowledge acquisition system. URL http://protege.stanford.edu/overview/protege-owl.html.

S. Pu and G. Vosselman. Automatic extraction of building features from terrestrial laser scanning. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36:25–27, 2006.

S. Pu and G. Vosselman. Knowledge based reconstruction of building models from terrestrial laser scanning data. *Journal of Photogrammetry and Remote Sensing*, 64(6): 575–584, 2009.

S Rajeev. *Artificial intelligence and expert systems for engineers*, volume 11. CRC press, 1996.

N. Ripperda and C. Brenner. Reconstruction of facade structures using a formal grammar and rjmcmc. *DAGM'06 proceedings of the 28th conference on Pattern Recognition*, pages 750–759, 2006.

U. Rost and H. Muenkel. Knowledge based configuration of image processing algorithms. *In Proceedings of the International Conference on Computational Intelligence and Multimedia Applications*, 1998.

R. B. Rusu, Z. C. Marton, N. Blodow, A. Holzbach, and M. Beetz. Model-based and learned semantic object labeling in 3d point cloud. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3601–3608, 2009.

Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

Heinrich Schewe, Jürgen Holl, and Lothar Gründig. Limez-photogrammetric measurement of railroad clearance obstacles. *Third Turkish-German Joint Geodetic Days, Istanbul/Turkey. Towards a Digital Age*, 2:721–727, 1999.

S. Scholze, T. Moons, Van Gool, and L. A probabilistic approach to building roof reconstruction using semantic labelling. *Pattern Recognition*, pages 257–264, 2002.

W3C semantic web. Resource description framework (rdf), 2004. URL http://www.w3.org/RDF.

W3C semantic web. Web ontology language (owl), 2007. URL http://www.w3.org/2004/OWL.

Shermarc. Why create artificial intelligence?, 2002. URL http://www.units.muohio.edu/psybersite/cyberspace/aisurge/implications.shtml.

L. Shi, S. Kodagoda, and G. Dissanayake. Laser range data based semantic labeling of places. *Intelligent Robots and Systems (IROS)*, pages 5941–5946, 2010.

Sylvie Soudarissanane, Roderik Lindenbergh, and Ben Gorte. Reducing the error in terrestrial laser scanning by optimizing the measurement set-up. *Proceedings of International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 3–11, 2008.

J. Stueckler and S. Behnke. Improving people awareness of service robots by semantic scene knowledge. *RoboCup 2010: Robot Soccer World Cup XIV. Lecture Notes in Computer Science*, 6556/2011:157–16, 2011.

Yuan Y Tang, Seong-Whan Lee, and Ching Y Suen. Automatic document processing: a survey. *Pattern Recognition*, 29(12):1931–1952, 1996.

Geoffrey Taylor and Lindsay Kleeman. Fusion of multimodal visual cues for model-based object tracking. *Conference on robotics and automation (ACRA)*, 2003.

Protege team. The semantic web rule language, 2012. URL http://protege.cim3.net/cgi-bin/wiki.pl?SWRLLanguageFAQ.

O. Teboul, L. Simon, Koutsourakis P., and N. Paragios. Segmentation of building facades using procedural shape priors. *Computer Vision and Pattern Recognition (CVPR)*, pages 3105–3112, 2010.

Trainmor. Basic knowledge concepts - data, information, knowledge and wisdom, 2013. URL http://www.trainmor-knowmore.eu/FBC5DDB3.en.aspx.

R. Triebel, O. M. Mozos, and W. Burgard. Relational learning in mobile robotics: An application to semantic labeling of objects in 2d and 3d environment maps. *in Annual Conference of the German Classification Society on Data Analysis, Machine Learning, and Applications (GfKl)*, 2007a.

R. Triebel, R. Schmidt, O. M. Mozos, and W. Burgard. Instace-based amn classification for improved object recognition in 2d and 3d laser range data. *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2007b.

John C. Trinder and Yandong Wang. Knowledge-based road interpretation in aerial images. *International Archives of Photogrammetry and Remote Sensing*, 32:635–640, 1998.

Hung Q. Truong, Sukhan Lee, and Seok-Woo Jang. Model-based recognition of 3d objects using intersecting lines. *In Multisensor Fusion and Integration for Intelligent Systems*, pages 656–660, 2008.

G Vosselman and S. Dijkman. 3d building model reconstruction from point clouds and ground plans. *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, 34:37–44, 2001.

M. Wuenstel and R. Moratz. Automatic object recognition within an office environment. *Proceedings of the 1st Canadian Conference on Computer and Robot Vision*, page 104–109, 2004.

Chaim Zins. Conceptual approaches for defining data, information, and knowledge. *Journal of the American Society for Information Science and Technology*, 58(4):479–493, 2007.

B Zitova and J. Flusser. Image registration methods: a survey. *Image and vision computing*, 21(11):977–1000, 2003.

List of Figures

List of Tables

Abstract:

The modeling of real-world scenes through capturing 3D digital data has proven to be both useful and applicable in a variety of industrial and surveying applications. Entire scenes are generally captured by laser scanners and represented by large unorganized point clouds possibly along with additional photogrammetric data. A typical challenge in processing such point clouds and data lies in detecting and classifying objects that are present in the scene. In addition to the presence of noise, occlusions and missing data, such tasks are often hindered by the irregularity of the capturing conditions both within the same dataset and from one data set to another. Given the complexity of the underlying problems, recent processing approaches attempt to exploit semantic knowledge for identifying and classifying objects. In the present thesis, we propose a novel approach that makes use of intelligent knowledge management strategies for processing of 3D point clouds as well as identifying and classifying objects in digitized scenes. Our approach extends the use of semantic knowledge to all stages of the processing, including the guidance of the individual data-driven processing algorithms. The complete solution consists in a multi-stage iterative concept based on three factors: the modeled knowledge, the package of algorithms, and a classification engine. The goal of the present work is to select and guide algorithms following an adaptive and intelligent strategy for detecting objects in point clouds. Experiments with two case studies demonstrate the applicability of our approach. The studies were carried out on scans of the waiting area of an airport and along the tracks of a railway. In both cases the goal was to detect and identify objects within a defined area. Results show that our approach succeeded in identifying the objects of interest while using various data types.

Keywords: 3D processing, point clouds, object detection, segmentation, algorithm selection, knowledge-based systems, knowledge modeling, ontology, classification

Résumé :

La modélisation de scènes réelles à travers la capture de données numériques 3D a été prouvée à la fois utile et applicable dans une variété d'applications. Des scènes entières sont généralement numérisées par des scanners laser et représentées par des grands nuages de points non organisés souvent accompagnés de données photogrammétriques. Un problème typique dans le traitement de ces nuages et données réside dans la détection et la classification des objets présents dans la scène. Ces tâches sont souvent entravées par la variabilité des conditions de capture des données, la présence de bruit, les occlusions ainsi que les données manquantes. Compte tenu de la complexité des problèmes sous-jacents, les approches de traitement récentes tentent d'exploiter les connaissances sémantiques pour identifier et classer les objets. Dans cette thèse, nous proposons une nouvelle approche qui fait appel à des stratégies intelligentes de gestion des connaissances pour le traitement des nuages de points 3D ainsi que l'identification et la classification des objets dans les scènes numérisées. Notre approche étend l'utilisation des connaissances sémantiques à toutes les étapes du traitement, y compris le choix et le guidage des algorithmes de traitement axées sur les données individuelles. Notre solution constitue un concept multi-étape itératif sur la base de trois facteurs: la connaissance modélisée, un ensemble d'algorithmes de traitement, et un moteur de classification. L'objectif de ce travail est de sélectionner et d'orienter les algorithmes de manière adaptative et intelligente pour détecter des objets dans les nuages de points. Des expériences avec deux études de cas démontrent l'applicabilité de notre approche. Les études ont été réalisées sur des analyses de la salle d'attente d'un aéroport et le long des voies de chemin de fer. Dans les deux cas, l'objectif était de détecter et d'identifier des objets dans une zone définie. Les résultats montrent que notre approche a réussi à identifier les objets d'intérêt tout en utilisant différents types de données.

Mots-clés : traitement 3D, nuages de points, détection d'objets, segmentation, sélection d'algorithme, systèmes basés connaissance, modélisation des connaissances, ontologies, classification

SPIM

■ École doctorale SPIM - Université de Bourgogne/UFR ST BP 47870 F - 21078 Dijon cedex
■ tél. +33 (0)3 80 39 59 10 ■ ed-spim@univ-fcomte.fr ■ www.ed-spim.univ-fcomte.fr

UNIVERSITÉ DE BOURGOGNE